UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Sistema de Recomendação de *frameworks* para Desenvolvimento Multiplataforma em Dispositivos Móveis

Dissertação de Mestrado

Denisson Santana dos Santos

São Cristóvão – Sergipe

2018

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Denisson Santana dos Santos

Sistema de Recomendação de *frameworks* para Desenvolvimento Multiplataforma em Dispositivos Móveis

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Prof. Dr. Alberto Costa Neto
Coorientador(a): Prof. Dr. Hendrik Teixeira Macedo

São Cristóvão – Sergipe

2018

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Santos, Denisson Santana dos
S237s Sistema de recomendação de *frameworks* para desenvolvimento multiplataforma em dispositivos móveis / Denisson Santana dos Santos ; orientador Alberto Costa Neto. - São Cristóvão, 2018.
107 f.; il.

Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Sergipe, 2018.

1. Ciência da computação. 2. Framework (Programa de computador). 3. Plataforma aberta da Web. 4. Aplicativos móveis. I. Costa Neto, Alberto. II. Título.

CDU 004.4

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: DENISSON SANTANA DOS SANTOS

Em 31 dias do mês de agosto do ano de dois mil e dezoito, com início às 14h00min, realizou-se na Sala de Reunião Superior do DCOMP da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Denisson Santana dos Santos**, que desenvolveu o trabalho intitulado: **" Sistema de Recomendação de frameworks para Desenvolvimento Multiplataforma em Dispositivos Móveis"**, sob a orientação do Prof. Dr. **Alberto Costa Neto**. A Sessão foi presidida pelo Prof. Dr. **Alberto Costa Neto** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Profª. Drª. **Adicinéia Aparecida de Oliveira** (PROCC/UFS) e, em seguida, ao Prof. Dr. **Rodrigo Bonifácio de Almeida** (UnB). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) APROVADO "(aprovado/reprovado)" SEM "(com/sem)" ressalvas. Atendidas as exigências da Instrução Normativa 01/2017/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 25/2014/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária *"Prof. José Aloísio de Campos"*, 31 de agosto de 2018.

Prof. Dr. Alberto Costa Neto
(PROCC/UFS)
Presidente

Profª. Drª. Adicinéia Aparecida de Oliveira
(PROCC/UFS)
Examinador Interno

Prof. Dr. Rodrigo Bonifácio de Almeida
(UnB)
Examinador Externo

Denisson Santana dos Santos
Candidato

UNIVERSIDADE FEDERAL DE SERGIPE
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO - PROCC
Sala do Seminário do DCOMP – Departamento de Computação / UFS, Av. Marechal Rondon, S/N - Jardim Rosa Elze - Tel. (79) 3194-6353. CEP: 49100-000 - São Cristóvão - Sergipe - Brasil
E-mail: secretaria.pos@dcomp.ufs.br

Resumo

O número e a variedade de dispositivos móveis têm crescido bastante atualmente, assim como o número de plataformas de diferentes fabricantes, sistemas operacionais e linguagens de programação existentes no mercado. Para acompanhar esse crescimento, os desenvolvedores, com o objetivo de disponibilizar suas aplicações para o maior número de plataformas, buscam cada vez mais otimizar a forma de desenvolver aplicativos *mobile*. Criar a mesma aplicação específica para cada plataforma demanda maior investimento financeiro, tempo e mão de obra. Para diminuir esses problemas surgiram os *frameworks* de desenvolvimento *mobile* para multiplataformas, onde torna-se possível a criação de aplicações para várias plataformas a partir do mesmo código fonte. No entanto, surge um dilema tanto para o desenvolvedor quanto para empresas de desenvolvimento: a pluralidade e diversidade desses *frameworks*, pois cada um possui objetivos e características que se aproximam e, em outros casos, distanciam-se um dos outros. Como por exemplo as diferentes linguagens usadas no desenvolvimento, as diferentes plataformas e recursos suportados, os diversos tipos de licença, dentre outros. Além disso, existe uma demanda por aplicativos de diversas finalidades e equipes de trabalho com diferentes perfis de formação. Por esses motivos apresentados, a escolha do *framework* não é trivial. Com o objetivo de ajudar os desenvolvedores e empresas de desenvolvimento de software *mobile*, este trabalho apresenta uma abordagem capaz de indicar qual *framework* de desenvolvimento de aplicativos multiplataforma é mais adequado para um projeto. Ademais, a abordagem supracitada foi implementada através de um sistema de recomendação que considerou os fatores críticos de um dado projeto informados pelo desenvolvedor.

Palavras-chave: recomendação, *framework*, multiplataforma, dispositivos móveis.

Abstract

The number and variety of mobile devices have increased so much nowadays, as well as the number of platforms on the market, from different manufacturers, operating systems and programming languages. To keep up with this increase, developers, aimming to make their applications available to the largest number of platforms, are increasingly seeking to optimize how to develop mobile applications. Create the same specific application for each platform demands greater financial, time and labor investment. In order to reduce these problems, mobile development frameworks for multiplatforms have emerged, making possible to create multi-platform applications from the same source code. However, a new issue arises for the developer - the plurality and diversity of these frameworks, each one with objectives and characteristics that sometimes make them closer and in other cases farther. In addition, there is a demand for multi-purpose applications and work teams with different training profiles. For these reasons, choosing a framework is not trivial. In order to help developers and companies of mobile software development, this paper presents a research proposal that consists of developing an approach that indicates which multi-platform application development framework is most suitable for a given project. In addition, the above mentioned approach was implemented through a knowledge-based recommendation system that considered the critical factors of a given project informed by the developer.

Keywords: recommendation, framework, cross-platform, mobile devices.

Lista de ilustrações

Lista de algoritmos

Lista de abreviaturas e siglas

4G	4ª Geração de Telefonia Móvel
API	*Application Programming Interface*
DPR	*Design Pattern Recommender*
GPS	*Global Positioning System*
IDE	*Integrated Development Environment*
MVC	*Model-View-Controller*
NFC	*Near Field Communication*
SO	Sistema Operacional
UFS	Universidade Federal de Sergipe

Sumário

1
Introdução

É notório que ao longo dos anos os dispositivos móveis passaram por algumas evoluções, e os fabricantes investiram grande quantidade de recursos financeiros a fim de modernizá-los e torná-los mais atrativos para os consumidores. Muitos foram os avanços pelos quais passaram (e continuarão a passar) tais dispositivos, a exemplo de uma melhor qualidade do sinal das operadoras (4G), precisão do *Global Positioning System* (GPS), variados tamanhos de tela, processadores poderosos, monitoramento de saúde, leitores de impressão digital, recursos de câmera, uso de realidade virtual, dentre outros. Atualmente, esses aparelhos são encontrados de diversas formas, como: *tablets*, *smartwatches*, *smartphones* e óculos. Ademais, com a crescente redução de preço desses aparelhos, é possível que pessoas com menor poder aquisitivo também tenham acesso a essa tecnologia.

Usualmente, no decurso da criação destes dispositivos os fabricantes adotam um sistema operacional (SO) ou, em alguns casos, optam por desenvolvê-lo. Com base nos dados da *International Data Corporation* (IDC) (2017) o Android, o iOS e o *Windows Phone* são os SOs para dispositivos móveis mais populares do mercado mundial. O Android lidera esse ranking e encontra-se em pleno crescimento, já que é possível utilizá-lo em aparelhos de fabricantes variados. O iOS, criado pela Apple, ocupa a segunda colocação, seguido pelo Windows Phone, que atinge a menor parcela do mercado. Logo, o desenvolvedor necessita desenvolver aplicativos nativos, ou seja, específicos para cada plataforma e linguagem, Objective-C ou Swift para iOS, Java para Android e C# para *Windows Phone*. Infelizmente, estas plataformas de desenvolvimento não têm muito em comum. Assim, demandam esforços de desenvolvimento em separado para entregar e manter um aplicativo corporativo para cada uma delas (GOKHALE; SINGH, 2014). O Quadro 1 resume bem as diferenças entre as plataformas supracitadas.

Quadro 1 – Principais plataformas de desenvolvimento.

Sistema Operacional	**Linguagem de Programação**	**Ambiente de Desenvolvimento**	**Loja de Aplicativos**
Android (Google)	JAVA	Android Studio	Play Store
iOS (Apple)	Objective-C / Swift	XCode	App Store
Windows Phone (Microsoft)	C#	Visual Studio	Windows Phone Market

Fonte – Latif et al. (2016).

Cada plataforma apresenta características peculiares, a saber, sistema operacional, conjunto de *frameworks* e bibliotecas, ferramentas de desenvolvimento e as linguagens de programação suportadas. Além disso, cada uma também possui suas lojas de aplicativos: *Play Store* (Android), *App Store* (iOS) e *Windows Phone Store (Windows Phone)*, as quais são utilizadas por desenvolvedores e empresas independentes que distribuem e vendem os aplicativos criados. Neste contexto, é interessante desenvolvê-los para o maior número possível de plataformas, a fim de alcançar mais usuários e assim maximizar o lucro. Porém, é preciso criar três versões do mesmo aplicativo com recursos completamente diferentes. Por essa razão, existe um crescente número de desenvolvedores e empresas que optam pelo uso do *cross-platform mobile app development framework* ou *frameworks* de desenvolvimento *mobile* para múltiplas plataformas.

Os *frameworks* de desenvolvimento *mobile* multiplataforma podem ser divididos em três tipos: *Webapps*; *Hibridos*; *Conversores*. As Webapps, também conhecidos como *frameworks* Web, permitem que desenvolvedores criem seus aplicativos através de tecnologias utilizadas para o desenvolvimento de sites e aplicações Web, tais como: HTML5, CSS3 e Javascript. Estes aplicativos são, na verdade, sites otimizados para o tamanho das telas dos dispositivos móveis e podem ser acessados por quaisquer plataformas (Android, iOS ou Windows Phone) através do navegador. Entretanto, são mais lentos, limitados ou simplesmente não podem explorar recursos de sistema operacional e, portanto, podem ficar aquém na experiência do usuário (GOKHALE; SINGH, 2014).

Os *frameworks* para desenvolvimento de aplicativos híbridos, tais como: Ionic e PhoneGap surgiram para tentar resolver alguns problemas das *Webapps*. Eles são parte nativos, parte *webapps*. Os *frameworks* híbridos criam uma base nativa do aplicativo e, dentro desta base, conseguem rodar uma página web. Por ter uma base nativa o *app* pode ser distribuído nas lojas e ter acesso a todos os recursos do dispositivo (MARKOV, 2015).

Para finalizar, os Conversores possuem uma linguagem de programação e bibliotecas próprias que o desenvolvedor utiliza para criar o aplicativo. O *framework* é responsável por converter o código desenvolvido para código nativo, específico para cada plataforma. Logo, o código é escrito uma única vez, em uma única linguagem de programação e convertido para as linguagens próprias de cada plataforma.

Neste contexto, este trabalho investigou quais são os principais *frameworks* de desenvolvimento de aplicativos multiplataforma e os fatores relevantes que envolvem a sua escolha dentre os vários existentes. Foi criado um questionário de pesquisa acerca desses fatores e aplicado em algumas comunidades de desenvolvimento mobile a fim de investigar a relevância de cada um deles na sua escolha. Em seguida, considerando os fatores definidos, foi desenvolvida uma abordagem que permite indicar qual *framework* de desenvolvimento de aplicativos multiplataforma é mais adequado a um dado desenvolvedor e seu projeto *mobile*.

Nas próximas seções deste capítulo, serão apresentados a justificativa para a realização deste trabalho, os objetivos gerais e específicos, as contribuições esperadas e a organização da dissertação.

1.1 Justificativa

O desenvolvimento de aplicativos para dispositivos móveis é uma atividade crítica, cuja realização se torna ainda mais difícil por conta da quantidade de plataformas, dispositivos e ferramentas existentes. O desenvolvimento nativo é a principal escolha, porém carrega algumas desvantagens, tais como: a necessidade de se escrever o mesmo código várias vezes (um para cada plataforma), logo o tempo e custo são acrescidos.

Deste modo, percebe-se a necessidade de abordagens que facilitem a criação de aplicações para plataformas diferentes utilizando o mesmo código-fonte. Para isso, há no mercado vários *frameworks* que oferecem suporte ao desenvolvimento multiplataforma. No entanto, devido a essa gama de opções, cada um com funcionalidades, características, linguagens e preços diferentes, sua escolha torna-se muitas vezes uma tarefa difícil, sobretudo para desenvolvedores que desejem experimentar essa forma de desenvolvimento pela primeira vez. As empresas não querem arriscar seus investimentos em qualquer ferramenta, tão pouco onerar desenvolvedores com longas provas de conceito sobre todas essas ferramentas presentes no mercado.

Durante a pesquisa realizada não foram encontradas soluções de recomendação deste objeto de estudo, tanto em ambiente acadêmico quanto em ambiente não acadêmico. Diante deste cenário, justifica-se a realização deste estudo que tem por finalidade fornecer uma abordagem que permita recomendar o *framework* de desenvolvimento *mobile* para múltiplas plataformas mais adequado ao desenvolvedor, com base em características informadas por ele sobre a equipe executora e sobre o projeto a ser desenvolvido.

1.2 Objetivos

O objetivo geral deste trabalho consiste em desenvolver e avaliar uma abordagem que indique qual *framework* de desenvolvimento de aplicativos multiplataforma mais se adéqua a

determinado desenvolvedor e seu respectivo projeto. Mais especificamente a um desenvolvedor com pouca ou nenhuma experiência em desenvolvimento *mobile* multiplataforma.

1.2.1 Objetivos Específicos

Para atender o objetivo principal, foram estabelecidos os seguintes objetivos específicos:

- Identificar os fatores que influenciam a escolha de um *framework* de desenvolvimento *mobile* para múltiplas plataformas.
- Implementar algoritmo de recomendação que considere os fatores que influenciam a escolha de um framework.
- Realizar um estudo de caso com o objetivo de aplicar a abordagem desenvolvida e implementada em um contexto real com foco nos desenvolvedores experientes.

1.3 Hipótese

É possível criar um sistema de recomendação baseado em conhecimento (*knowledge-based*), onde o conhecimento será representado por fatores críticos relacionados aos *frameworks* de desenvolvimento *mobile* multiplataforma pré-selecionados. O sistema deverá considerar os fatores supracitados a fim de conseguir recomendar o *framework* que mais se adéque.

1.4 Metodologia de Pesquisa

A natureza de pesquisa é Pura, pois este trabalho contribui para a evolução da própria Computação, mais especificamente da área de Engenharia de software, utilizando ferramental de inteligência artificial. Quanto aos objetivos, a pesquisa é exploratória, pois visa proporcionar maior familiaridade com o problema a fim de torná-lo mais explícito ou a desenvolver hipóteses (SILVA; MENEZES, 2001). Quanto aos procedimentos técnicos, estudo de caso é o procedimento mais adequado, pois conduz a análise do caso de modo detalhado com o objetivo de adquirir conhecimento a partir do fenômeno estudado (VENTURA, 2007).

Para este trabalho foi estabelecido o seguinte conjunto de etapas:

- Revisão da literatura a fim de encontrar pesquisas semelhantes à desenvolvida neste trabalho.
- Identificação dos principais *frameworks mobile* multiplataforma utilizados pelo mercado.
- Identificação dos critérios de escolha de um *frameworks mobile* multiplataforma através de um survey aplicado na comunidade de desenvolvimento de aplicativos móveis.

- Implementação da abordagem de recomendação mais adequada para o cenário estudado nesta pesquisa.
- Validação da abordagem de recomendação através de um survey aplicado na comunidade de desenvolvimento de aplicativos móveis.

1.5 Organização da Dissertação

Os demais capítulos desta dissertação são relacionados a seguir:

- O Capítulo 2 apresenta a fundamentação teórica, onde são apresentados os conceitos que fornecem base ao desenvolvimento deste trabalho, destacando-se: abordagens de recomendação; *frameworks* selecionados para estudo; medidas de similaridade; e tecnologias envolvidas.
- O Capítulo 3 apresenta os trabalhos relacionados com este estudo. Através dessas publicações será possível extrair abordagens, conceitos e fundamentos para que ao final desse trabalho tenha-se desenvolvido um projeto bem embasado.
- O Capítulo 4 apresenta e discute os principais fatores que caracterizam a escolha mais adequada para um determinado projeto, todos levantados através de pesquisa realizada por meio de questionário online disponibilizado em listas de discussão especializadas em desenvolvimento *mobile*.
- O Capítulo 5 detalha a abordagem construída para a escolha de um *framework* de desenvolvimento de aplicativos multiplataforma que mais se adéque a fatores relacionados à equipe de desenvolvimento e ao projeto a ser desenvolvido. Também apresenta o sistema de recomendação que implementa a abordagem desenvolvida.
- O Capítulo 6 avalia o desempenho da abordagem de recomendação desenvolvida, considerando os fatores que foram definidos no capítulo 4.
- O Capítulo 7 apresenta de forma resumida o trabalho desenvolvido, as principais contribuições, sugestões de trabalhos futuros e as conclusões deste trabalho.

2

Desenvolvimento para Dispositivos Móveis

No presente capítulo são apresentados os conceitos teóricos fundamentais acerca do desenvolvimento para dispositivos móveis. Na primeira seção, 2.1, são apresentadas as abordagens de desenvolvimento *mobile* multiplataforma. Em seguida, na seção 2.2, são expostos os principais *frameworks* de desenvolvimento *mobile* multiplataforma utilizados nesse trabalho.

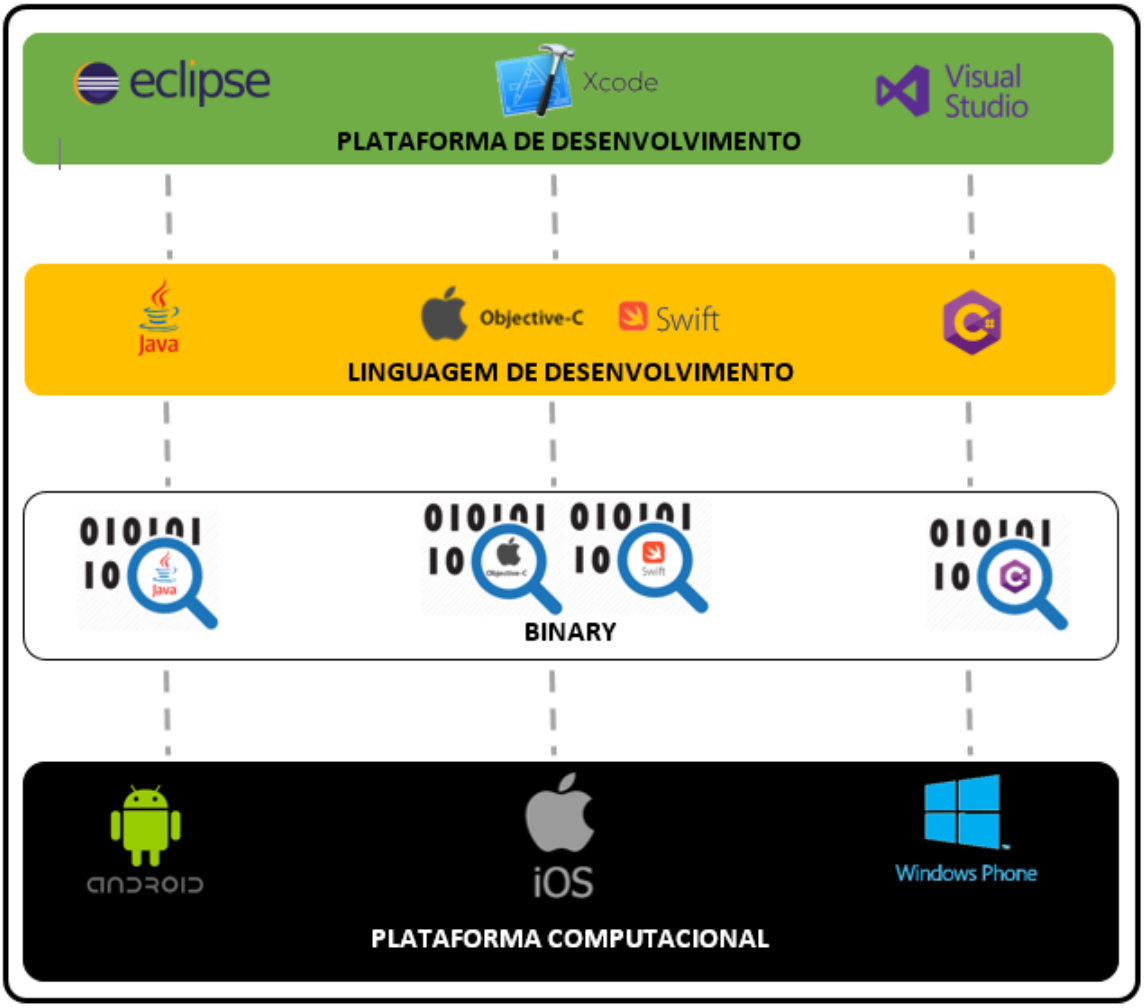
2.1 Abordagens de Desenvolvimento *Mobile*

De acordo com El-Kassas et al. (2017), o desenvolvimento de aplicativos *mobile* possui quatro abordagens: Desenvolvimento Nativo, o qual utiliza linguagem e ferramental próprio; Desenvolvimento Web, que utiliza linguagem web para desenvolver aplicativos que rodarão em navegadores; Desenvolvimento Híbrido, que cria uma base nativa do aplicativo e, dentro desta base, conseguem rodar uma página web; e Desenvolvimento Multiplataforma, onde o aplicativo será desenvolvido em uma determinada linguagem e, em seguida, convertida para código nativo.

2.1.1 Desenvolvimento Nativo

O desenvolvimento nativo é voltado para a criação de aplicações específicas para uma determinada plataforma, como: iOS, Android, *Windows Phone*, ou seja, geram binários específicos e utilizam linguagens e muitas vezes ambientes de desenvolvimento próprio, como é ilustrado na Figura 1.

Figura 1 – Abordagem Nativa.



Fonte - Latif et al. (2016).

Nessa abordagem, a interface gráfica seguirá o padrão que o usuário já está habituado. Além disso, a interação de funcionalidades com o dispositivo, como câmera e GPS, apresenta melhor desempenho quando comparado com as aplicações mobile multiplataforma (FOWLER, 2012). No entanto, esse tipo de abordagem torna difícil a reusabilidade, pois cada plataforma utiliza uma linguagem de programação diferente das demais. Geralmente, torna-se necessário montar vários times de desenvolvimento, de modo que cada um é especializado em uma plataforma específica. Assim, observa-se que essa abordagem aumenta consideravelmente os custos, pois além de implementar várias versões da aplicação, será necessário manutenir todas elas. Logo, criar uma nova funcionalidade ou alterá-la significa desenvolver e testar em cada uma das diferentes plataformas de desenvolvimento.

2.1.2 Desenvolvimento Web

O desenvolvimento web consiste em criar um aplicativo através do uso de tecnologias de desenvolvimento Web, tais como: HTML[1], JavaScript[2] e CSS[3] e distribui-la para qualquer plataforma, ou seja, desenvolvimento totalmente independente da plataforma. (BERNARDES; MIYAKE, 2016) Este tipo de desenvolvimento proporciona a criação de aplicativos usando menos recursos, como: banda larga e espaço de armazenamento dos dispositivos. Ele faz uso do navegador web, assim, não há necessidade de efetuar nenhuma instalação no dispositivo do usuário. Essa abordagem reduz significativamente o tempo de desenvolvimento, pois, além de permitir a criação e manutenção de apenas uma aplicação, utiliza tecnologias amplamente conhecidas da Web, e, deste modo, evita-se a necessidade de conhecimentos específicos de diferentes plataformas.

Na contramão das vantagens apresentadas, existem algumas desvantagens em usá-los, tais como: os aplicativos criados através dessa abordagem geralmente não conseguem interagir com alguns atributos do aparelho, como: acelerômetro, câmera, GPS, entre outros (GOKHALE; SINGH, 2014); dependem de internet, não utilizam toda a capacidade gráfica do dispositivo e não podem ser disponibilizados em lojas de app (BERNARDES; MIYAKE, 2016).

As webApps tentam se aproximar da experiência de uso de um app nativo com um custo de implementação inferior. Por serem pensados objetivando apenas dispositivos móveis, eles tendem a ter boas interfaces, se comparados a sites responsivos. Apesar de não ser possível acessar alguns recursos do dispositivo, esse tipo de desenvolvimento fornece uma grande vantagem em relação aos apps nativos, quando a grande questão diz respeito à implantação em várias plataformas.

2.1.3 Desenvolvimento Híbrido

O desenvolvimento híbrido de aplicações é realizado através das abordagens nativa e web simultaneamente, a fim de alcançar praticidade na entrega do produto final em multiplataforma. Neste contexto, o desenvolvimento é construído com base em tecnologias web, e, através de um *framework* para desenvolvimento de aplicativos móveis multiplataformas, o código é encapsulado em uma base nativa (C.P; TOLETY, 2012).

Em seguida, o código é renderizado em um componente web, onde será carregado automaticamente pela base nativa. Esta mesma base interage com o sistema operacional e com o hardware da plataforma, assim, torna-se possível acessar recursos que os aplicativos web são incapazes de ter acesso. Dessa forma, não haverá impedimento no desenvolvimento de funcionalidades que necessitem de acesso às funções do dispositivo, a exemplo da câmera e do GPS.

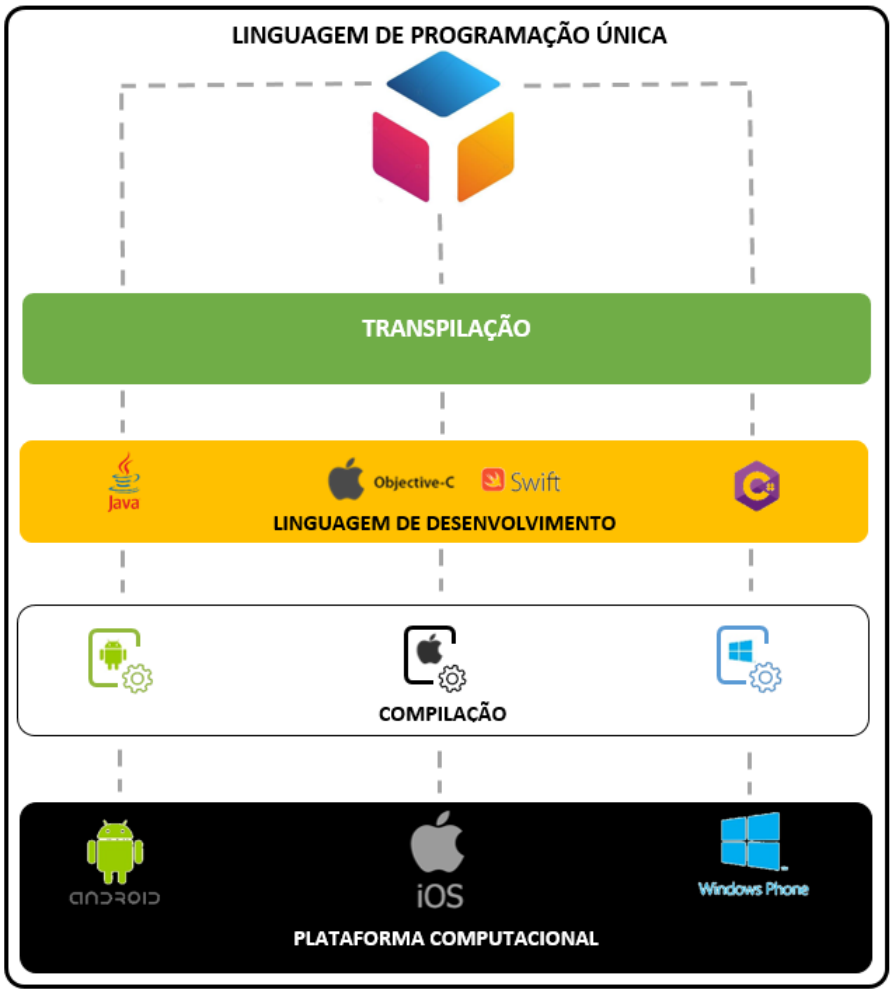
[1] <http://www.html.com/>
[2] <https://www.javascript.com/>
[3] <http://www.w3schools.com/css/>

Por fim, os aplicativos híbridos são mais facilmente distribuídos aos usuários em relação aos *web*, pois possuem uma base nativa, logo, podem ser disponibilizados pelas lojas das plataformas. Com o aumento do uso desta abordagem, vários *Frameworks* para desenvolvimento de aplicativos móveis multiplataformas (será detalhado na próxima seção) foram criados. Muitos deles diferenciam-se um dos outros pelos seus atributos (funcionalidades, preço, linguagens, entre outros) e objetivos. É nesse ponto que este trabalho se estabelece, pois ele se propõe a sugerir ao desenvolvedor qual deles melhor se adéqua ao projeto que se deseja construir.

2.1.4 Desenvolvimento Multiplataforma

De acordo com El-Kassas et al. (2017), as soluções multiplataforma ajudam os desenvolvedores a escrever o código-fonte uma vez em determinada linguagem, transpilar esse código para código nativo e disponibilizar o aplicativo móvel produzido em diferentes plataformas, conforme ilustra a Figura 2.

Figura 2 – Abordagem Multiplataforma.



Fonte - Latif et al. (2016).

O Xamarin, por exemplo, *framework* da Microsoft, permite que os aplicativos sejam

escritos em C#, mas quando o app for gerado para iOS, por exemplo, será realizada uma transpilação do código para Objective-C, então um projeto nativo é disponibilizado. O mesmo ocorre para aplicativos desenvolvidos para Android, em que o código C# é transpilado para Java. Ademais, caso seja desejo do desenvolvedor, ele poderá seguir no projeto desenvolvendo nativamente.

Similar ao que ocorre na seção anterior, esse tipo de desenvolvimento se torna possível por meio dos *frameworks mobile* multiplataforma, que serão discutidos na próxima seção.

2.2 *Frameworks* para Desenvolvimento de Aplicativos Móveis Multiplataformas

De acordo com Boushehrinejadmoradi et al. (2015), os desenvolvedores de aplicativos para dispositivos móveis geralmente desejam disponibilizar seus aplicativos em uma grande variedade de plataformas, a saber - dispositivos Android, iOS e *Windows*. Cada uma dessas plataformas usa um ambiente de programação diferente, cada um com seu próprio idioma e APIs para desenvolvimento de aplicativos. Pequenas equipes de desenvolvimento de aplicativos não têm os recursos e a experiência necessária para criar e manter bases de código separadas do aplicativo personalizado para cada plataforma. Esse cenário abriu espaço para o surgimento de uma série de *frameworks* de desenvolvimento de aplicativos móveis para múltiplas plataformas. Esses *frameworks* permitem que os desenvolvedores de aplicativos especifiquem a lógica de negócios do aplicativo uma vez, usando a linguagem e as APIs suportadas por eles e produzam automaticamente versões do aplicativo para várias plataformas de destino (por exemplo, iOS e Android). Desenvolver um único código e distribuir para várias plataformas é um ponto importante não só para as pequenas equipes, como mencionado acima, mas para empresas que visam economia de tempo e custo. É preciso destacar que a construção desses *frameworks* é desafiadora, pois eles devem traduzir corretamente o que foi desenvolvido através deles para a plataforma destino, ao mesmo tempo que devem fornecer o mesmo comportamento de uma aplicação desenvolvida nativamente.

Os vários *frameworks* existentes são caracterizados por vários fatores, dentre eles estão: licença, linguagem que é utilizada para desenvolver os aplicativos e recursos suportados (GPS, bússola, NFC, etc) (BERNARDES; MIYAKE, 2016). Um dos objetivos deste trabalho é descobrir quais os principais fatores que implicam na escolha ou não de determinado *framework*. Alguns fatores são determinantes nesse processo de definição de qual *framework* é o mais indicado. Por exemplo, existe *framework* que gera aplicativos para iOS e Android, mas não gera para Windows Phone. Caso o desenvolvedor queira que seu app seja disponibilizado nas três plataforma, ele deverá escolher um que entregue para as três.

2.2.1 Requisitos para o desenvolvimento mobile multiplataforma através de *frameworks*

Segundo Latif et al. (2016), o principal objetivo dos *frameworks* é produzir aplicativos móveis para múltiplas plataformas, mas é importante levar em consideração os requisitos necessários para a produção e manutenção de aplicações móveis de alto desempenho. Latif et al. (2016), através de análise própria, identificaram alguns dos requisitos desejáveis para qualquer tecnologia multiplataforma, segue abaixo:

- **Manutenibilidade**: O desenvolvimento mobile multiplataforma deve possibilitar manter e melhorar a aplicação. Por exemplo, se novas funcionalidades precisam ser implementadas em um aplicativo, as modificações devem sempre estar no nível multiplataforma e, em seguida, implantadas em plataformas específicas.

- **Facilidade de acesso a dispositivos**: O aplicativo criado usando multiplataforma deve ter acesso aos recursos do dispositivo, e, ainda, que seja preferivelmente um acesso direto sem nenhuma camada intermédia.

- **Consumo de recursos**: Os desenvolvedores precisam levar em consideração o consumo de recursos, incluindo CPU, consumo de energia e uso de memória. Mas não só os desenvolvedores devem se preocupar com essa questão, os *frameworks* multiplataformas devem oferecer máxima otimização desses recursos no momento da implantação na plataforma específica.

- **Segurança**: A questão da segurança é um dos critérios mais importantes no desenvolvimento móvel. Parte desse problema pode ser resolvida pelo desenvolvedor. O desenvolvedor deve prestar atenção aos dados armazenados no dispositivo, pois os dispositivos podem ser roubados ou cair em mãos não autorizadas. No entanto, os *frameworks* multiplataformas devem ter a capacidade de respeitar as políticas de segurança definidas pelos desenvolvedores em todos os diferentes sistemas operacionais.

- **Ambiente de desenvolvimento**: O ambiente de desenvolvimento é um fator muito importante quando se fala em *frameworks* multiplataformas, ele deve integrar todos os depuradores e compiladores necessários. Mas o fator mais importante são os simuladores para diferentes sistemas (Android, iOS, Windows Phone, dentre outros), com eles é possível obter uma boa visibilidade do aplicativo antes da implantação.

2.2.2 Principais *Frameworks*

Este trabalho selecionou os principais *frameworks* do mercado encontrados na revisão sistemática (BERNARDES; MIYAKE, 2016), que também figuraram nas listas dos *frameworks* mais populares segundo os sites (KHAN, 2015; TIWARI, 2016; GIASSON, 2017; SINGH,

2017; FURLAN, 2017; REDBYTES, 2017; MEDIUM, 2017; JSCRAMBLER, 2017), e que possuem uma alta popularidade no Github[4] e no Stackoverflow[5]. Além deles, suas principais características foram pesquisadas em sites oficiais e ambos estão descritos abaixo.

- **Xamarin**[6]: É um *framework* multiplataforma que possibilita ao desenvolvedor criar aplicativos, através da linguagem C#, com controle de interface de usuário padrão nativo. O Xamarin aproveita a aceleração de hardware específica da plataforma e compila os apps para desempenho nativo. Isso não pode ser alcançado com soluções que interpretam o código em tempo de execução. Os aplicativos Xamarin têm acesso a todo o espectro de funcionalidades expostas pela plataforma e pelo dispositivo subjacentes, incluindo recursos específicos da plataforma. O Xamarin é mantido pela Microsoft, que o distribui através de alguns planos: pagos e gratuito. Abaixo estão suas principais características:

  Plataformas Suportadas: iOS, Android e Windows Phone
  Linguagem suportada: C#
  Tipo de aquisição ($): Flexível (Existem opções pagas e gratuitas)
  Licença: Proprietária
  Compras internas: Sim
  Envio do app para loja: Sim
  Recursos Suportados: Bússola, GPS, Acelerômetro, 3D Touch, Assistente Virtual, NFC, Leitor de Impressão Digital e Aceleração de Vídeo
  Geração de *Bytecodes*: Sim
  Suporte a versões mais recentes das plataformas suportadas: Imediato
  Atualizações Constantes do *Framework*: Sim
  Editor Gráfico: Sim
  IDE - Integrated Development Environment: Sim

- **Ionic**[7]: É um *framework* open source para desenvolvimento de aplicativos móveis multiplataforma que utiliza o *framework* Apache Cordova como base. Como diferencial o Ionic promete trazer recursos que simplificam ainda mais o desenvolvimento e dar ao app um aspecto mais profissional. Para desenvolver apps o desenvolvedor utiliza tecnologia web, a saber: HTML, CSS e TypeScript. Essa é uma vantagem para aqueles programadores que tem conhecimento em tecnologias web. Abaixo estão suas principais características:

  Plataformas Suportadas: iOS, Android e Windows Phone
  Linguagem suportada: TypeScript
  Tipo de aquisição ($): Flexível (Existem opções pagas e gratuitas)
  Licença: *Open-Source (MIT License)*

[4] <http://github.com>
[5] <https://stackoverflow.com/>
[6] <https://xamarin.com/>
[7] <https://ionic.io/>

Compras internas: Sim
Envio do app para loja: Não
Recursos Suportados: Bússola, GPS, Acelerômetro, 3D Touch, Assistente Virtual, NFC, Leitor de Impressão Digital e Aceleração de Vídeo
Geração de *Bytecodes*: Não
Suporte a versões mais recentes das plataformas suportadas: Imediato
Atualizações Constantes do *Framework*: Sim
Editor Gráfico: Sim
IDE - Integrated Development Environment: Não

- **Phonegap**[8]: Ele permite ao desenvolvedor aproveitar seus conhecimentos adquiridos no desenvolvimento web para criar rapidamente aplicativos híbridos criados com HTML, CSS e JavaScript. Dessa forma, é possível criar distribuições do mesmo app para várias plataformas com uma única base de código. Isso implica no alcance de maior público, pois seu app será independente de dispositivo e plataforma. A estrutura do PhoneGap, mantida pela Adobe, é composta do código aberto do Cordova que fornece a vantagem da tecnologia criada por uma equipe diversificada de profissionais, juntamente com uma robusta comunidade de desenvolvedores. Abaixo estão suas principais características:

  Plataformas Suportadas: iOS, Android e Windows Phone
  Linguagem suportada: Javascript + HTML
  Tipo de aquisição ($): Grátis
  Licença: *Open-Source (MIT License)*
  Compras internas: Sim
  Envio do app para loja: Não
  Recursos Suportados: Bússola, GPS, Acelerômetro, 3D Touch, Assistente Virtual, NFC, Leitor de Impressão Digital e Aceleração de Vídeo
  Geração de *Bytecodes*: Não
  Suporte a versões mais recentes das plataformas suportadas: Imediato
  Atualizações Constantes do *Framework*: Sim
  Editor Gráfico: Não
  IDE - Integrated Development Environment: Não

- **Appcelerator Titanium**[9]: É um *framework* open-source que permite ao desenvolvedor criar aplicativos móveis em todas as plataformas a partir de uma única base de código JavaScript e API Titanium. O Titanium permite que você desenvolva um app para várias plataformas com menor tempo, mas com os mesmos resultados da plataformas nativas. O Titanium gera aplicativos verdadeiramente nativos. Para isso, ele fornece uma camada responsável pro transpilar o código Javascript do desenvolvedor para código nativo de

[8] <http://phonegap.com/>
[9] <http://appcelerator.com/>

cada plataforma. Segundo os criadores este framework é apoiado por uma comunidade de desenvolvimento de mais de 660.000 desenvolvedores móveis com 75.000 aplicativos móveis implantados em mais de 280.000.000 de dispositivos. Abaixo estão suas principais características:

Plataformas Suportadas: iOS e Android
Linguagem suportada: Javascript
Tipo de aquisição ($): Flexível (Existem opções pagas e gratuitas)
Licença: *Open-Source (Apache 2 License)*
Compras internas: Sim
Envio do app para loja: Não
Recursos Suportados: Bússola, GPS, Acelerômetro, 3D Touch, Assistente Virtual, NFC, Leitor de Impressão Digital e Aceleração de Vídeo
Geração de *Bytecodes*: Sim
Suporte a versões mais recentes das plataformas suportadas: Imediato
Atualizações Constantes do *Framework*: Sim
Editor Gráfico: Sim
IDE - Integrated Development Environment: Sim

- **Corona**[10]:É um *framework* multiplataforma open-source, totalmente gratuito, ideal para criar aplicativos, especialmente jogos para dispositivos móveis. Isso significa que é possível desenvolver (na linguagem LUA) um projeto uma única vez e publicá-lo nas duas principais plataformas: Android (smartphones e tablets) e iOS (iPhone e iPad). Abaixo estão suas principais características:

  Plataformas Suportadas: iOS e Android
  Linguagem suportada: LUA
  Tipo de aquisição ($): Grátis
  Licença: Proprietária
  Compras internas: Sim
  Envio do app para loja: Sim
  Recursos Suportados: Bússola, GPS, Acelerômetro, 3D Touch, NFC e Aceleração de Video
  Geração de *Bytecodes*: Sim
  Suporte a versões mais recentes das plataformas suportadas: Imediato
  Atualizações Constantes do *Framework*: Sim
  Editor Gráfico: Não
  IDE - Integrated Development Environment: Não

- **Sencha**[11]: É um *framework* multiplataforma que oferece ao desenvolvedor web maior

[10] <https://coronalabs.com/>
[11] <https://sencha.com>

facilidade no desenvolvimento de apps, pois utiliza tecnologia web, tais como: HTML, Javascript e CSS. Ele gera aplicações para Android, iOS e Windows Phone. A companhia Sencha, proprietária deste framework, o distribui sob duas licenças: Comercial, onde o desenvolvedor precisa pagar pelo uso, após um período de experimentação e outra gratuita, onde o usuário pode desenvolver com a condição de disponibilizar todo código produzido. Abaixo estão suas principais características:

Plataformas Suportadas: iOS, Android e Windows Phone
Linguagem suportada: Javascript + HTML
Tipo de aquisição ($): Flexível (Existem opções pagas e gratuitas)
Licença: *Open-Source (MIT License)*
Compras internas: Sim
Envio do app para loja: Não
Recursos Suportados: Bússola, GPS, Acelerômetro e Assistente Virtual
Geração de *Bytecodes*: Não
Suporte a versões mais recentes das plataformas suportadas: Imediato
Atualizações Constantes do *Framework*: Não
Editor Gráfico: Não
IDE - Integrated Development Environment: Sim

- **Rhodes**[12]: É uma solução de código aberto para a criação de aplicativos *mobile* multiplataformas. Com ele é possível criar aplicativos utilizando a linguagem Ruby juntamente com HTML e Javascript e distribuir para as plataformas: iOS, Android e Windows Phone. Os desenvolvedores que conhecem o *framework* Rails terão vantagem pois o Rhodes utiliza padrões de desenvolvimento semelhantes, por exemplo o MVC. Outra vantagem se apresenta para os desenvolvedores web, pois estes podem aproveitar seus conhecimentos e aplicar no desenvolvimento de apps neste *framework*. Abaixo estão suas principais características:

  Plataformas Suportadas: iOS, Android e Windows Phone
  Linguagem suportada: Ruby, HTML e Javascript
  Tipo de aquisição ($): Flexível (Existem opções pagas e gratuitas)
  Licença: *Open-Source (MIT License)*
  Compras internas: Sim
  Envio do app para loja: Não
  Recursos Suportados: Bússola, GPS, Acelerômetro, Assistente Virtual, NFC e Aceleração de Vídeo
  Geração de *Bytecodes*: Sim
  Suporte a versões mais recentes das plataformas suportadas: De 3 a 6 meses
  Atualizações Constantes do *Framework*: Sim

[12] <http://tau-technologies.com/products/rhomobile/>

Editor Gráfico: Não
IDE - Integrated Development Environment: Sim

- **React Native**[13]: Com o React Native, é possível criar, de forma gratuita, a partir de um único código programado em javascript com a biblioteca react, um aplicativo móvel multiplataforma, que segundo o Facebook, criador e mantenedor do *framework*, é igual a um aplicativo criado usando o Objective-C ou o Java. React Native combina componentes escritos em Objective-C, Java ou Swift, assim, é simples descer para o código nativo se você precisar otimizar alguns aspectos do seu aplicativo. Também é fácil criar uma parte do seu aplicativo no React Native e outra usando código nativo diretamente - é assim que funciona o aplicativo do Facebook. Abaixo estão suas principais características:

  Plataformas Suportadas: iOS, Android e Windows Phone
  Linguagem suportada: Javascript + React
  Tipo de aquisição ($): Grátis
  Licença: *Open-Source (MIT License)*
  Compras internas: Sim
  Envio do app para loja: Não
  Recursos Suportados: Bússola, GPS, Acelerômetro, 3D Touch, Assistente Virtual, NFC, Leitor de Impressão Digital e Aceleração de Vídeo
  Geração de *Bytecodes*: Não
  Suporte a versões mais recentes das plataformas suportadas: Imediato
  Atualizações Constantes do *Framework*: Sim
  Editor Gráfico: Não
  IDE - Integrated Development Environment: Não

No Quadro 2 é apresentada uma visão geral dos *frameworks* analisados juntamente com as características que os definem. Em seguida, no Quadro 3, são apresentados também os recursos por eles suportados.

[13] <http://reactnative.com/>

Quadro 2 – Características dos Frameworks Analisados.

	React Native	**Ionic**	**Phonegap**	**Corona**	**Xamarin**	**Titanium**	**Sencha**	**Rhodes**
Plataformas Suportadas	Android, iOS e Windows Phone	Android, iOS e Windows Phone	Android, iOS e Windows Phone	Android e iOS	Android, iOS e Windows Phone	Android e iOS	Android, iOS e Windows Phone	Android, iOS e Windows Phone
***IDE* própria**	Não	Não	Não	Sim	Sim	Sim	Sim	Sim
Linguagem suportada	Javascript + React	TypeScript	Javascript + HTML	LUA	C#	Javascript	Javascript + HTML	Ruby e Javascript + HTML
Tipo de aquisição ($)	Grátis	Flexível	Grátis	Grátis	Flexível	Flexível	Flexível	Grátis
Licença	Open-Source (MIT License)	Open-Source (MIT License)	Open-Source (MIT License)	Proprietária	Proprietária	Open-Source (Apache 2 License)	Open-Source (MIT License)	Open-Source (MIT License)
Geração de *Bytecodes*	Não	Não	Não	Sim	Sim	Sim	Não	Sim
Suporte a versões mais recentes das plataformas	Imediato	Imediato	Imediato	Imediato	Imediato	Imediato	Imediato	3 a 6 meses
Envio do app para loja	Não	Não	Não	Sim	Sim	Não	Não	Não
Compras internas	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Atualizações Constantes	Sim	Sim	Sim	Sim	Sim	Sim	Não	Sim
Editor Gráfico	Não	Sim	Não	Não	Sim	Sim	Não	Não

Quadro 3 – Recursos Suportados.

	React Native	**Ionic**	**Phonegap**	**Corona**	**Xamarin**	**Titanium**	**Sencha**	**Rhodes**
Bússola	X	X	X	X	X	X	X	X
GPS	X	X	X	X	X	X	X	X
Acelerômetro	X	X	X	X	X	X	X	X
3D Touch	X	X	X	X	X	X		
Assistente Virtual	X	X	X		X	X	X	X
NFC	X	X	X	X	X	X		X
Leitor de Impressão Digital	X	X	X		X	X		
Aceleração de Vídeo	X	X	X	X	X	X		X

Os frameworks e suas características expostas nos Quadros acima compõem a base de conhecimento utilizada na abordagem de recomendação baseada em conhecimento. Na próxima seção serão introduzidos os conceitos sobre sistema de recomendação, os quais forneceram subsídios teóricos para o desenvolvimento da abordagem proposta neste trabalho que será discutida posteriormente no capítulo 5.

2.3 Considerações Finais do Capítulo

Ao longo deste capítulo foram apresentados conceitos acerca dos tipos de desenvolvimento de aplicativos *mobile* utilizados pelos desenvolvedores móveis, assim como suas vantagens e desvantagens. Foi discutido também sobre os *frameworks mobile* multiplataforma cujo objetivo é possibilitar a distribuição de um aplicativo em diversas plataformas gerados a partir do mesmo código fonte. Também foram identificados os principais *frameworks* utilizados pelo mercado, com o objetivo de considerá-los na abordagem de recomendação que será apresentada no capítulo 5.

O próximo capítulo apresentará os sistemas de recomendação e seus principais tipos, além dos trabalhos que serviram de base para o desenvolvimento desta pesquisa.

3

Sistemas de Recomendação

A crescente importância da Web como um meio para transações eletrônicas e comerciais tem sido de suma importância para o desenvolvimento da tecnologia de sistemas de recomendação. Isso ocorre, pois com a grande rede de computadores é possível colher a opinião dos usuários acerca do que gostam ou não, de diversas formas, direta ou indiretamente. A Netflix[1], famosa plataforma provedora de filmes e séries via *streaming*, tem o *feedback* do usuário simplesmente considerando as opções clicadas por ele, ou quando este avalia alguma série ou filme através do sistema de classificação fornecido pela plataforma. A Amazon.com usa outra forma também muito comum de receber *feedback*, que se dá pelo simples ato de um usuário comprar ou navegar em um item de uma loja virtual (AGGARWAL, 2016b).

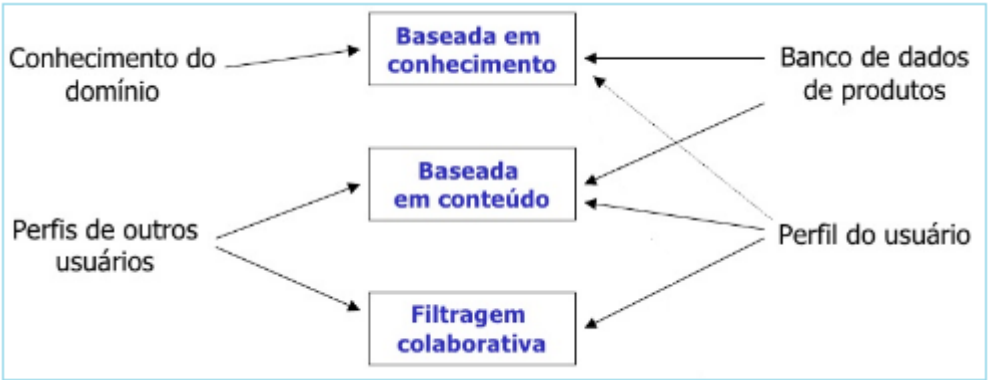
De posse dos interesses do usuário ou características informadas por ele, as plataformas podem recomendar de forma personalizada os melhores conteúdos, itens ou produtos para ele, pois, de acordo com Kumar, Tanwar e Nigam (2016), os sistemas de recomendação são desenvolvidos com o objetivo de ajudá-los a obter escolhas personalizadas a fim de facilitar a obtenção de informações relevantes na internet. Para realizar essa tarefa, muitos modelos de aprendizagem diferentes podem ser usados, dentre eles estão: filtragem colaborativa, filtragem baseada em conteúdo e filtragem baseada no conhecimento, Figura 3.

Um modelo bastante conhecido é a filtragem colaborativa. O comportamento coletivo de compra ou avaliação de vários usuários, pode ser o ponto chave para criar grupos de usuários similares que estão interessados em produtos semelhantes. Dessa forma os interesses e ações desses grupos serão insumos para que recomendações sejam feitas aos membros individuais deles (AGGARWAL, 2016b). De acordo com Kumar, Tanwar e Nigam (2016), a correlação de Pearson pode ser usada para pesar a similaridade do usuário com todos os vizinhos correlatos e a previsão final pode ser encontrada calculando a média ponderada de desvio da média do vizinho:

$p_{a,i}$ descreve a previsão do usuário a para o item i, n é o número de vizinhos e $w_{a,u}$ é

[1] <https://netflix.com/br/>

Figura 3 – Modelos de Aprendizagem.



$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n}(r_{u,i} - \bar{r}_u) * w_{a,u}}{\sum_{u=1}^{n} w_{a,u}}$$

o peso de similaridade entre o usuário a e o vizinho u, conforme definido pelo coeficiente de correlação de Pearson:

$$w_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \bar{r}_a) * (r_{u,i} - \bar{r}_u)}{\sigma_a * \sigma_u}$$

Outra abordagem bastante conhecida são os sistemas de recomendação baseados em conteúdo. O conteúdo desempenha um papel primordial no processo de recomendação, no qual os interesses e preferências do usuário são cruzados com os atributos dos itens, descrições e características que ele interagiu anteriormente, dessa forma, novos itens lhes serão recomendados (LOPS; SEMERARO, 2011). Um bom exemplo é a recomendação de filmes. Quando um determinado usuário assiste e avalia bem um dado filme, outros filmes desse mesmo gênero lhe serão recomendados.

Um quadro diferente é o dos sistemas baseados no conhecimento, nos quais as recomendações são sugeridas com base em requisitos especificados pelo usuário e não no histórico passado por ele (BURKE, 2000). Em outras palavras, os usuários interativamente especificam seus interesses, e a especificação deles é combinada com o conhecimento do domínio para fornecer recomendações (AGGARWAL, 2016b). Por exemplo, um recomendador de restaurantes utiliza essa abordagem, então espera-se que o usuário forneça características como: tipo de comida, preço, local, dentre outros. Com essas informações o sistema recomenda uma lista de restaurantes por ordem de similaridade (BURKE, 2000). É importante destacar que em modelos avançados, dados contextuais, como informações temporais, conhecimento externo, informações de localização, informações sociais ou informações de rede, também podem ser usados (AGGARWAL, 2016b). Essa abordagem é a que mais se adéqua ao trabalho proposto, pois o que é desejado é que *frameworks mobile* multiplataforma sejam sugeridos de acordo com

características (preço, linguagem de programação, finalidade do projeto, etc) do desenvolvedor e da aplicação a ser desenvolvida, informadas pelo desenvolvedor. A seguir será detalhado esse tipo de sistema de recomendação.

3.1 Sistema de Recomendação Baseado em Conhecimento

Os sistemas de recomendação baseados no conhecimento são particularmente úteis no contexto de itens que não são procurados com muita frequência ou acabaram de ser lançados, logo, não possuem classificações suficientes para o processo de recomendação. Em outros casos, pode ser difícil capturar completamente o interesse do usuário com dados históricos, como classificações. Um item em particular pode ter atributos associados a ele que correspondem às suas várias propriedades, e um usuário pode estar interessado apenas em itens com propriedades específicas. Por exemplo, o objeto deste trabalho - *frameworks mobile* multiplataforma pode ter vários atributos, como: preço, linguagem, licença, recursos suportados, entre outros, e os interesses do usuário podem ser uma combinação muito específica dessas opções. Assim, nestes casos, o domínio do item tende a ser complexo em termos das suas propriedades variadas, e é difícil associar classificações suficientes ao grande número de combinações existentes. Tais casos podem ser abordados com sistemas de recomendação baseados em conhecimento.

O processo de recomendação é realizado com base em semelhanças entre as necessidades dos clientes e as descrições dos itens, ou o uso de restrições que especificam os requisitos do usuário. O processo é facilitado com o uso de bases de conhecimento, que contêm dados sobre regras e funções de similaridade a serem usadas durante o processo de recuperação. A especificação explícita dos requisitos resulta em maior controle dos usuários sobre o processo de recomendação. Tanto em sistemas colaborativos como em sistemas baseados em conteúdo, as recomendações são decididas inteiramente pelas ações passadas do usuário, pela ação / classificação de seus pares ou por uma combinação dos dois. Os sistemas baseados no conhecimento são únicos no sentido de permitir que os usuários especifiquem explicitamente o que desejam (AGGARWAL, 2016b).

Os sistemas de recomendação baseados em conhecimento são classificados em:

1. Sistemas de Recomendação Baseados em Restrições: Nesses sistemas os usuários especificam requisitos ou restrições acerca dos atributos do item. As regras específicas, também conhecidas como conhecimento do domínio, são usadas para corresponder aos requisitos do usuário aos atributos do item. Além disso, os sistemas baseados em restrições geralmente criam regras relacionando atributos de usuário com atributos de item, por exemplo, "Homens obesos não consomem verduras". Nesses casos, os atributos do usuário também podem ser especificados no processo de pesquisa. Dependendo do número e tipo de resultados retornados, o usuário pode ter a oportunidade de modificar seus requisitos

originais. Este processo de pesquisa é interativamente repetido até que o usuário chegue aos resultados desejados (AGGARWAL, 2016a).

2. Sistemas de Recomendação Baseados em Casos: Essa abordagem é uma forma de sugestão baseada em conteúdo que pode ser bem adaptada a vários domínios de sugestões de produtos, onde produtos separados são denominados com base num conjunto exato de características, tais como: preço, cor, marca, região, etc. A estas características são aplicadas métricas de similaridade para determinar semelhança entre os itens com a finalidade de melhorar a recomendação. Verificou-se extremamente eficaz em vários ambientes de negócios *online*, especialmente quando os desejos e gostos dos clientes são ambíguos e vagos (SINGAL; TEJAL; JUNEJA, 2016).

   Para que um sistema de recomendação baseado em casos seja eficiente, existem dois pontos importantes do sistema que devem ser destacados:

   a) Métricas de similaridade: Para avaliar a similaridade de dois itens, as similaridades para todos os atributos individuais são calculadas aplicando funções de similaridade local a cada par de parâmetros correspondentes. Estas semelhanças de parâmetros locais são usadas para calcular um valor de similaridade geral para os dois dispositivos. Uma das formas de se calcular similaridade geral é utilizando a média ponderada das semelhanças individuais. Os fatores de peso podem ser sugeridos por especialistas no negócio a ser implementado, como também ser influenciados pelo cliente (VOLLRATH; WILKE; BERGMANN, 1998). Por exemplo, nesse trabalho, os *frameworks mobile* multiplataforma, desenvolvedores e objetivos do projeto possuem atributos de diferentes tipos. Para saber se um *framework* possui suporte a mapas, a resposta é sim ou não, já para saber o preço que um desenvolvedor deseja pagar, a resposta é um valor numérico. Dessa forma, tipos diferentes de métricas de similaridade devem ser combinados para que essa abordagem seja aplicada corretamente. (CHA, 2007) apresenta várias medidas para diversas ocasiões. Segundo Vollrath, Wilke e Bergmann (1998), encontrar uma medida de similaridade adequada é muitas vezes a parte mais crítica da concepção e implementação de um sistema de recomendação baseado em casos. Uma vez que uma boa medida de similaridade foi encontrada e implementada, a manutenção e desenvolvimento do sistema torna-se bastante fácil. Para esse trabalho a similaridade entre os *frameworks* pode ser alcançada através da distância euclidiana ponderada, pois cada atributo dispõe de um nível de importância (peso):

$$D(X,Y) = (\frac{1}{\sum iw_i})^{\frac{1}{2}} * [\sum iw_i * dist(X_i,Y_i)^2]^{\frac{1}{2}}$$

   onde 0 <= w_i <= 1 (peso), i = 1..n (número de atributos), $dist(X_i,Y_i)$ é a medida de distância específica para cada tipo de atributo, que deve estar normalizado. Por

exemplo, nas plataformas suportadas pelo *framework* (Android, iOs e Windows Phone), pode-se usar a medida de Jaccard (AGGARWAL, 2016a), que é dada por:

$$dist(X_i, Y_i) = 1 - \frac{a}{a+b+c}$$

onde:

a = número de ocorrências da plataforma escolhida e presentes no *framework*.

b = número de ocorrências da plataforma escolhida e ausentes no *framework*.

c = número de ausências da plataforma escolhida e presentes no *framework*.

Existe uma variedade de medidas de distância apropriadas para as várias situações que se queira medir (conjunto de booleanos, hierarquia, enumerados, número, etc). Cha (2007) cita várias medidas de distância / similaridade que são aplicáveis para comparar conjuntos de atributos dos mais variados formatos, tais como: booleano, numérico, hierárquico, multivalorados, entre outros.

b) Métodos de crítica: Críticas são motivadas pelo fato dos usuários muitas vezes não estarem em condições de indicar os seus requisitos na consulta inicial. Em alguns domínios complexos, eles podem até achar difícil traduzir suas necessidades de forma semântica significativa para os valores de atributos no domínio do produto. É somente depois de ver os resultados de uma consulta que um usuário pode perceber que ele deve formular sua consulta de forma um pouco diferente. Em outras palavras, nas críticas, os usuários especificam solicitações de alteração na consulta em um ou mais atributos de um item que eles podem gostar. Críticas são projetados para fornecer aos usuários essa habilidade após o fato. (AGGARWAL, 2016a).

3.2 Fragilidades

Os sistemas de recomendação também apresentam fragilidades que podem comprometer a probabilidade de acerto das sugestões dos itens aos usuários, são elas:

- *Cold-start*: Quando se trata de usuários e itens novos, os sistemas de recomendação enfrentam um grande desafio, pois não é possível encontrar usuários ou itens similares para servir de base para realizar a recomendação. Além dos usuários e itens novos existem os itens obscuros, ou seja, aqueles que raramente alguém escolhe, isso acaba prejudicando aqueles com gostos mais alternativos. Este problema é mais comum na abordagem de filtragem colaborativa e por conteúdo (LAVETI et al., 2016).

- *Over-Specialization*: Este problema é mais comum na abordagem de filtragem por conteúdo, pois o sistema pode não recomendar nada além do perfil do usuário, que muitas vezes não é atualizado. Dessa maneira o sistema termina ignorando as evoluções quanto aos interesses do usuário à medida que o tempo passa (JAIN et al., 2015).

- Fraude: Alguns sistemas são mais penalizados por conta de fraude de concorrentes, os quais estão em busca da valorização de seus produtos e desvalorização dos produtos de outras marcas. Pessoas podem criar perfis falsos para impactar diretamente na recomendação através da atribuição de notas baixas para o concorrente e altas para seus produtos, dessa forma seus produtos poderão ter lucros maximizados e os produtos da concorrência terão uma baixa probabilidade de serem escolhidos. Este problema é mais comum na abordagem de filtragem colaborativa (JAIN et al., 2015).

- Esparsidade: A filtragem colaborativa necessita da avaliação dos usuários para que seja possível realizar sugestões para usuários similares, no entanto, vários usuários podem não avaliar o sistema, por diversos motivos: falta de tempo, esquecimento ou por optarem não avaliar. Essas atitudes geram uma matriz de avaliação muito esparsa. Como resultado, a probabilidade de encontrar usuários com avaliações similares é significativamente reduzida. Este problema é mais comum na abordagem de filtragem colaborativa (JAIN et al., 2015).

- Privacidade: A depender do propósito da recomendação, o fator privacidade pode ser empecilho para o usuário fazer uma avaliação. O receio de fornecer informações como idade, localização, data de aniversário, sexo, email, dentre outros, causa sério prejuízo na coleta de dados, assim as sugestões não são feitas com a devida precisão (JAIN et al., 2015).

A próxima seção apresentará os principais trabalhos utilizados para entender e fundamentar os conceitos utilizados para desenvolver a abordagem de recomendação de *frameworks mobile* multiplataforma.

3.3 Trabalhos Relacionados

Foi realizada uma pesquisa nas bases *IEEE Xplorer Digital Library*[2], *Springer*[3], *Google Scholar*[4] e *ACM Digital Library*[5] a fim de encontrar trabalhos que apresentassem estudos sobre abordagens de indicação de *frameworks* para desenvolvimento *mobile* multiplataforma. Foram utilizados os seguintes termos de busca: *"cross-platform mobile"*, *"development cross-platform"*, *"recommender development mobile"*, *"recommendation development mobile"*, *"recommender system cross-platform"*, *"recommendation system cross-platform"*, *"recommendation development cross-platform"*, *"recommender development cross-platform"*, *"recommendation mobile framework"*, *"recommender mobile framework"*, *"recommender system framework"* e *"recommendation system framework"*. Para buscar trabalhos que fundamentassem os sistemas de

[2] <http://ieeexplore.ieee.org/Xplore/home.jsp>
[3] <https://link.springer.com/>
[4] <https://scholar.google.com>
[5] <http://dl.acm.org/>

recomendação, e assim ter um melhor entendimento sobre o assunto foram utilizados os seguintes termos de busca:*"recommender system"* e *"recommendation system"*. Para ter um melhor entendimento sobre medidas de similaridades foi utilizado o seguinte termo de busca:*"Similarity Measures"*. As buscas foram realizadas com período definido entre 2007 e 2017, ou seja, nos últimos 10 anos. Após a análise dos títulos dos artigos não foi encontrado nenhum artigo que tratasse especificamente a cerca de sugestão de *framework mobile* multiplataforma. Apesar disso, foram detectados artigos que utilizam algoritmos de recomendação para outras finalidades e um artigo que faz uma comparação entre alguns *frameworks mobile* multiplataforma. Além desses, foram encontrados outros artigos com potencial para auxiliar no entendimento de sistemas de recomendação e métricas de similaridade. Os trabalhos são:

- Bernardes e Miyake (2016) realizaram uma revisão sistemática com o objetivo de obter uma comparação entre as abordagens de desenvolvimento multiplataforma, e assim identificar os pontos fortes e fracos de cada uma. Nesse estudo os autores também listaram vários *frameworks* utilizados no mercado tecnológico. Essa informação é muito importante para este trabalho, pois um dos objetivos específicos é justamente definir uma lista atualizada dos frameworks mais usados pelos desenvolvedores.

- Palmieri, Singh e Cicchetti (2012) realizaram uma comparação entre quatro *frameworks* de desenvolvimento *mobile* multiplataformas que no mínimo gere *apps* para Android, iOS, BlackBerry e Windows Phone, a saber, PhoneGap, Rhodes, DragonRad e MoSync, com o objetivo de auxiliar desenvolvedores na escolha do *framework* mais adequado às suas necessidades. Foram comparadas funcionalidades, ambiente de desenvolvimento, licenças, interfaces, plataformas e linguagens de programação suportadas. Estes itens comparados são bastante relevantes para integrar a lista de critérios que foi utilizada na abordagem proposta neste trabalho.

- Jain et al. (2015) apresentam em seu trabalho definições das principais abordagens de sistemas de recomendação, a saber, filtragem por conteúdo; filtragem colaborativa; filtragem demográfica; filtragem híbrida. Para demonstrar a popularidade dos sistemas de recomendação, os autores apresentam importantes empresas que fazem uso dessa abordagem, tais como: LinkedIn[6] e Amazon[7]. Este artigo também apresenta possíveis desafios que podem ser encontrados a partir da utilização da técnica de recomendação, como: *Cold-start*; *Over-Specialization*; Escalabilidade; Esparsidade; Previsibilidade; Privacidade. O estudo desses conceitos foi fundamental para aprimorar o conhecimento sobre sistemas de recomendação, o que possibilitou desenvolver o sistema proposto neste trabalho com maior embasamento.

[6] <https://linkedin.com/>
[7] <https://amazon.com/>

- Burke (2000) introduz conceitos que compõem a abordagem de sistema de recomendação conhecida como *Knowlegde-Based recommender systems*. Nesse artigo o autor apresenta um pequeno histórico através de exemplos aplicados a sistemas clássicos desse nicho, como o sistema *FindMe*, mostra também conceitos de similaridade e métricas. O artigo também aborda os sistemas híbridos, que combina a abordagem citada com a *Colaborative filtering*. Esse artigo mostra-se muito importante tanto por abordar conceitos que serviram de base para o trabalho desenvolvido, quanto na apresentação de exemplos práticos, assim, muito sobre o funcionamento e a própria interface dão dicas sobre a construção de um novo sistema.

- Aggarwal (2016a) apresenta de forma bastante detalhada os conceitos acerca de uma das abordagens de sistemas de recomendação conhecida como *Knowlegde-Based recommender systems*. Ele se divide em *Case-Based* e *Constraint-Based*, onde este se baseia em restrições duras e regras predefinidas e aquele considera regras de similaridade como filtro. A similaridade entre atributos pode ser tratada através de alguns modelos matemáticos, como expõe este artigo. Essa abordagem de sistema de recomendação, em especial o tipo *Case-Based* mostra-se bastante adequada ao trabalho desenvolvido, pois o foco deste trabalho é encontrar o framework que mais se aproxime dos atributos informados pelo desenvolvedor, não estritamente igual.

- Cha (2007) apresenta uma *survey* onde são apresentados vários tipos de medidas de distância ou similaridade. Elas são essenciais para resolver muitos problemas que tem solução em sistemas de recomendação baseado em conhecimento, especialmente na classificação *Case-based*. O autor cita várias medidas de distância / similaridade que são aplicáveis para comparar conjuntos de atributos dos mais variados formatos, como: booleano, numérico, hierárquico, multivalorados, entre outros. No trabalho desenvolvido foram comparados vários tipos de atributos pertencentes aos *frameworks*, a fim de determinar o nível de similaridade entre eles. Para cada um foi utilizada uma métrica apropriada, assim, esse artigo foi de grande importância para este trabalho.

Os trabalhos acima expostos serviram de base para o desenvolvimento da abordagem de recomendação de *frameworks mobile* multiplataforma. Como explicado acima, cada um oferece pontos que contribuíram para que o trabalho esteja bem fundamentado e direcionado.

3.4 Considerações Finais do Capítulo

Neste capítulo os sistemas de recomendação e seus principais tipos foram abordados, com destaque para a recomendação baseada em conhecimento, a qual foi escolhida para ser utilizada na abordagem desenvolvida. Também foram alvos deste capítulo, os trabalhos que serviram de base para o desenvolvimento desta pesquisa.

O próximo capítulo apresentará um *survey* que foi distribuído em listas de discussão compostas por desenvolvedores *mobile*, cujo objetivo foi identificar os principais critérios para a escolha de um *framework mobile* multiplataforma.

4

Levantamento de Fatores Determinantes na Adoção de *Frameworks* Mobile Multiplataforma

Este capítulo descreve a pesquisa realizada para ajudar a definir os fatores mais relevantes para a adoção do *framework mobile* multiplataforma mais adequado a um dado desenvolvedor e projeto. Para isso, foi criado um questionário de pesquisa *online* com o intuito de investigar quais seriam esses fatores. As informações coletadas foram essenciais para a definição dos critérios de escolha dos *frameworks* que foram considerados no sistema de recomendação baseado em conteúdo. Além disso, foi possível entender através do questionário de pesquisa o quão experientes são os participantes em relação ao uso de projetos que envolvem *frameworks mobile* multiplataforma. Logo, pode-se ter mais segurança do quão coerente são os fatores de adoção apontados pelos participantes da pesquisa.

De acordo com Pfleeger e Kitchenham (2001), um questionário de pesquisa figura como um sistema abrangente de coleta de informações para descrever, comparar ou explicar conhecimento, atitude ou comportamento. Neste sentido, este instrumento faz parte de um amplo processo, com atividades bem definidas, tais como: definir objetivos específicos e quantificáveis; projetar o questionário; selecionar os participantes; metodologia para análise; análise dos resultados.

4.1 Definição de Objetivos Específicos e Quantificáveis

A *survey* foi criada em meio eletrônico através da ferramenta TypeForm[1], com o objetivo de coletar dados a fim de investigar sobre os fatores que influenciam no processo de adoção de *frameworks mobile* multiplataforma. Foram definidos 2 objetivos específicos e quantificáveis, segue:

[1] <https://www.typeform.com/>

- Definir o nível de importância dos fatores que influenciam na decisão de escolha de *frameworks mobile* multiplataforma;
- Avaliar se todos os fatores pré-selecionados devem ser considerados na definição de critérios de avaliação a serem utilizados no desenvolvimento do sistema de recomendação baseado em conteúdo.

4.2 Projeto do Questionário

O questionário foi elaborado com questões que se dividem em três grupos: informações pessoais, informação sobre a experiência do perguntado e informações tecnológicas acerca dos *frameworks*.

O primeiro grupo de perguntas trata sobre as informações pessoais do participante, a fim de conhecer um pouco melhor sobre quem está respondendo as questões. Os participantes foram questionados com as seguintes perguntas:

1. Qual a sua idade?
2. Qual seu gênero?
3. Qual seu país de nascimento?
4. Qual sua situação profissional?
5. Qual sua escolaridade?

O segundo grupo de questões trata acerca do nível de experiência que os participantes têm na área de desenvolvimento *mobile*. Esse grupo de questões foi elaborado com o objetivo de certificar que os perguntados vivenciam ambientes que lidam com este tipo de desenvolvimento, pois a relevância desse *survey* depende da expertise dos perguntados. Seguem as perguntas deste segundo grupo:

1. Você possui alguma experiência com desenvolvimento de aplicativos para dispositivos móveis?
2. Quais linguagens de programação você conhece?
3. Legal que você conhece (Linguagem escolhida na pergunta anterior)! Qual seu nível de conhecimento na linguagem?
4. Você estaria disposto a aprender uma nova linguagem de programação para poder desenvolver aplicativos mobile?

5. Você (ou sua empresa) já desenvolveu algum aplicativo mobile para mais de uma plataforma (Android, iOS, etc)?
6. Para quais plataformas mobile você ou sua empresa costuma desenvolver?
7. Você (ou sua empresa) já criou algum aplicativo utilizando algum framework de desenvolvimento mobile multiplataforma?
8. Quais fatores te levaram (ou levariam) a escolher um framework de desenvolvimento multiplataforma?
9. Você já conhece algum framework de desenvolvimento de aplicativos multiplataforma?
10. Quais dos Frameworks de desenvolvimento multiplataforma abaixo você conhece?

Por fim, o terceiro grupo de perguntas trata sobre o objetivo principal deste *survey* – descobrir a relevância que alguns fatores têm ou não para cada desenvolvedor no momento da escolha de um *framework mobile* multiplataforma. Abaixo seguem as perguntas elaboradas para este terceiro grupo:

1. Na sua opinião, quais as plataformas mais importantes do mercado?
2. Que tipo de abordagem de desenvolvimento mobile você considera ideal?
3. Quais recursos do dispositivo um framework precisa suportar para satisfazer suas necessidades atuais?
4. Para você, dos itens abaixo qual o mais importante no momento de escolher um framework?
5. Você pagaria para utilizar um framework multiplataforma?
6. O quão importante é o suporte à versão mais recente da plataforma (Android, iOS) no momento da escolha de um framework multiplataforma?

Juntos, os três grupos de perguntas fornecem informações importantes para a definição da lista de fatores que serão considerados pelo sistema de recomendação, em especial porque será possível descobrir o nível de experiência das pessoas que responderão o *survey* na área de desenvolvimento *mobile*, logo teremos mais segurança na aceitação das respostas.

4.3 Seleção dos Participantes

Para que se tenha uma amostra válida de perguntados, é necessário definir as pessoas para quem o questionário é voltado. Para definir uma amostra dessa população alvo existem dois métodos, segundo Pfleeger e Kitchenham (2002): Probabilístico e Não-Probabilístico. O método

probabilístico se aplica ao caso onde todos os membros da população pesquisada são conhecidos e apresentam uma probabilidade maior que zero para ser incluída na amostra; enquanto que no método não-probabilístico os participantes da pesquisa são incluídos na amostra porque são facilmente acessíveis ou os pesquisadores entendem que eles são representativos para a população alvo.

O método não-probabilístico foi escolhido. Assim, neste trabalho, o questionário de pesquisa foi divulgado abertamente em relevantes listas de discussão voltadas essencialmente para desenvolvedores *mobile*. Seguem as listas:

- iOSDevBrasil;
- iOSDev-br;
- AndroidDev-br;
- Android-Brasil.

4.4 Metodologia para Análise

Os resultados que envolvem algumas questões podem apresentar algumas superposições, ou seja, se as porcentagens dessas respostas forem somadas, o resultado será superior a 100%, pois algumas perguntas permitiram respostas com múltiplas escolha. Algumas respostas relacionadas diretamente a fatores que influenciam na adoção de um *frameworks mobile* multiplataforma seguiram uma escala de influência que vai de 1 - Pouco Importante, passa por 3 - Média importância e vai até 5 - Extremamente Importante. A análise foi feita considerando especialmente o número de respostas que variaram de Importante a Extremamente Importante. Desta forma, foi possível selecionar os fatores que foram considerados relevantes o suficiente para uma avaliação coerente deste projeto.

4.5 Análise dos Resultados

O questionário ficou disponível na internet para que os participantes respondessem às perguntas por um período de 2 meses. Após esse período foram obtidas 136 respostas.

Os participantes desta pesquisa, em sua maioria, eram profissionais da área de TI com nível superior, como pode ser observado nas Figuras 4 e 5. Na Figura 5 Empregado, Empresário e Servidor Público somam 83% e na Figura 4 Graduação, Pós-graduação, Mestrado e Doutorado somam 86%. A experiência em desenvolvimento *mobile* dos participantes pode ser vista nas Figuras 6, 7, 8, 9, 10 e 11. A Figura 7 mostra que os participantes possuem experiência com desenvolvimento de aplicativos para dispositivos móveis, pois todas as resposta que vão de 3 - médio até 5 - *Expert* totalizaram 83%; 70% dos perguntados já desenvolveram para mais de

uma plataforma *mobile*, Figura 6; Os 95 participantes que se encaixam no 70% que responderam que desenvolveram para mais de uma plataforma também responderam em sua maioria que desenvolveram para Android, iOS e Windows Phone, apenas 2% desenvolveu para Blackberry, como pode ser visto na Figura 8; A Figura 9 mostra que 78% dos participantes conhecem algum *frameworks mobile* multiplataforma e 51% dos questionados já desenvolveram aplicativos por meio deles, como ilustrado na Figura 11. Na Figura 10 ainda é possível conhecer a popularidade dos *frameworks* selecionados entre os participantes. Neste contexto, as características desta amostra permite assumir que os participantes desta pesquisa possuem maturidade para ajudar a definir quais fatores são relevantes para a adoção de *frameworks mobile* multiplataforma.

Figura 4 – Escolaridade.

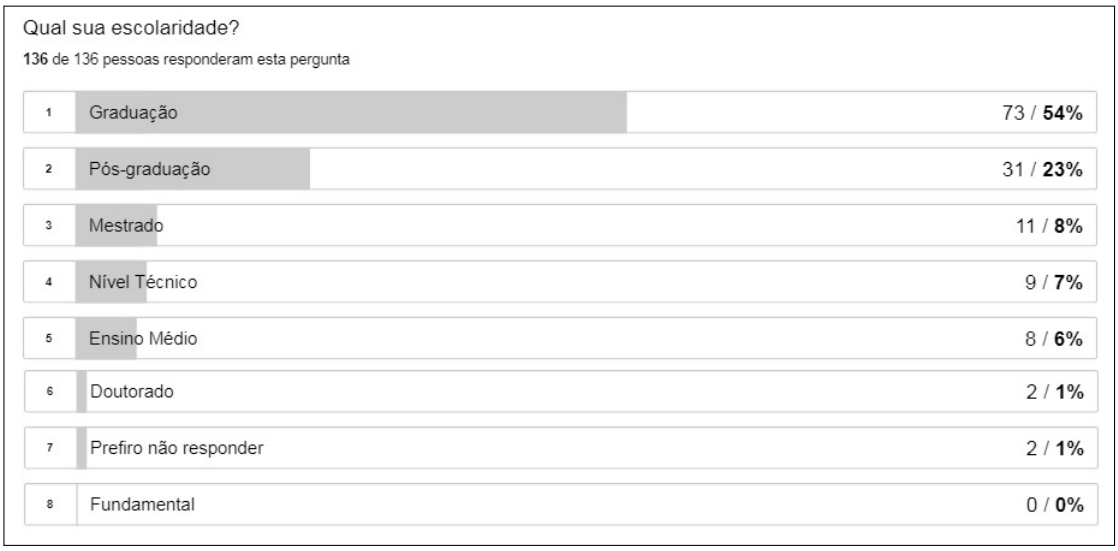


Figura 5 – Situação Profissional.

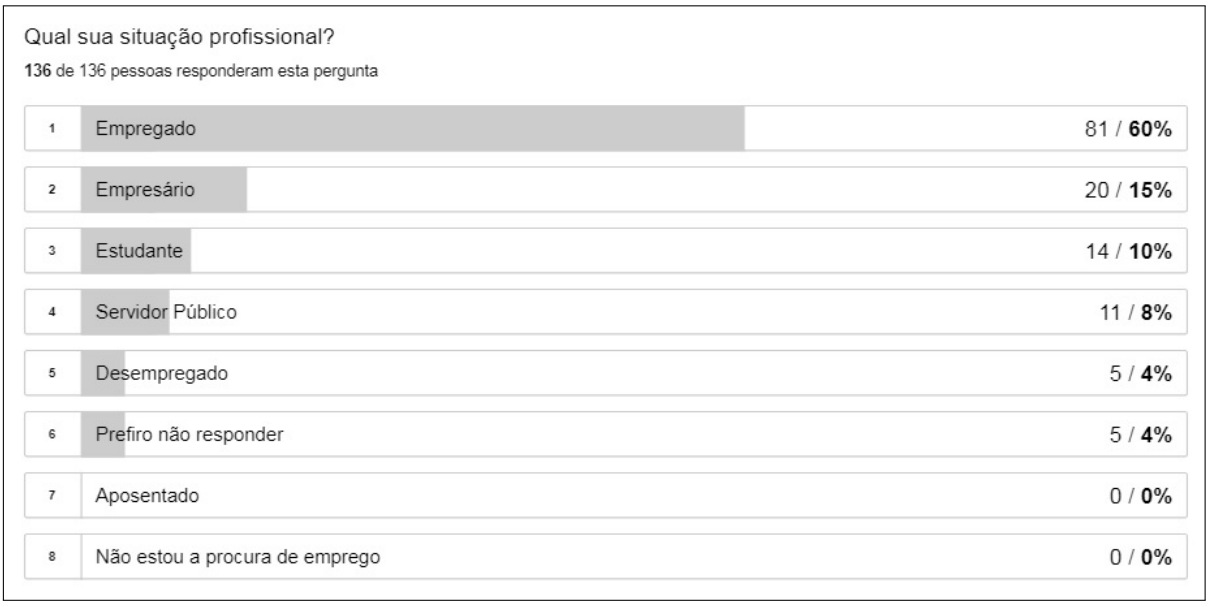


Figura 6 – Experiência com desenvolvimento para várias plataformas.

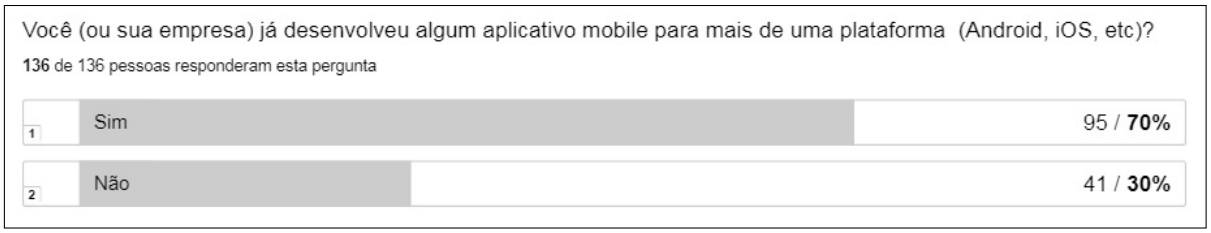


Figura 7 – Experiência com desenvolvimento *mobile*.

Figura 8 – Plataforma para a qual mais se desenvolve.

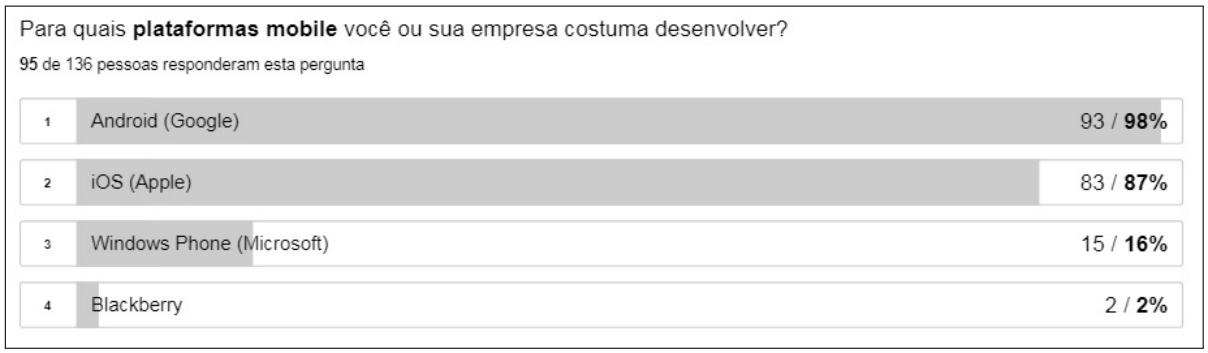


Figura 9 – Conhecimento acerca de *frameworks mobile* multiplataforma.

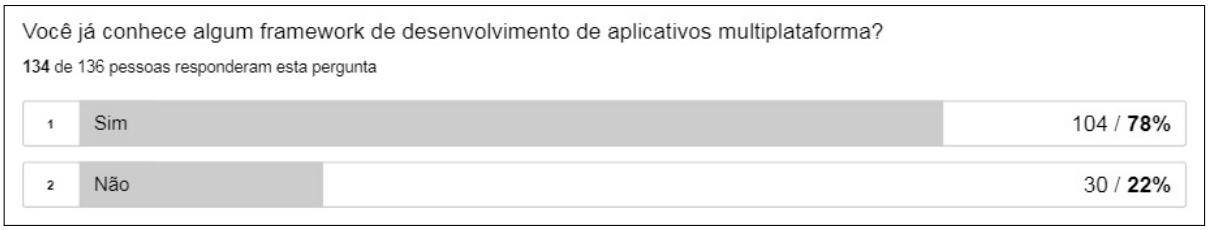


Figura 10 – *Frameworks mobile* multiplataforma mais populares.

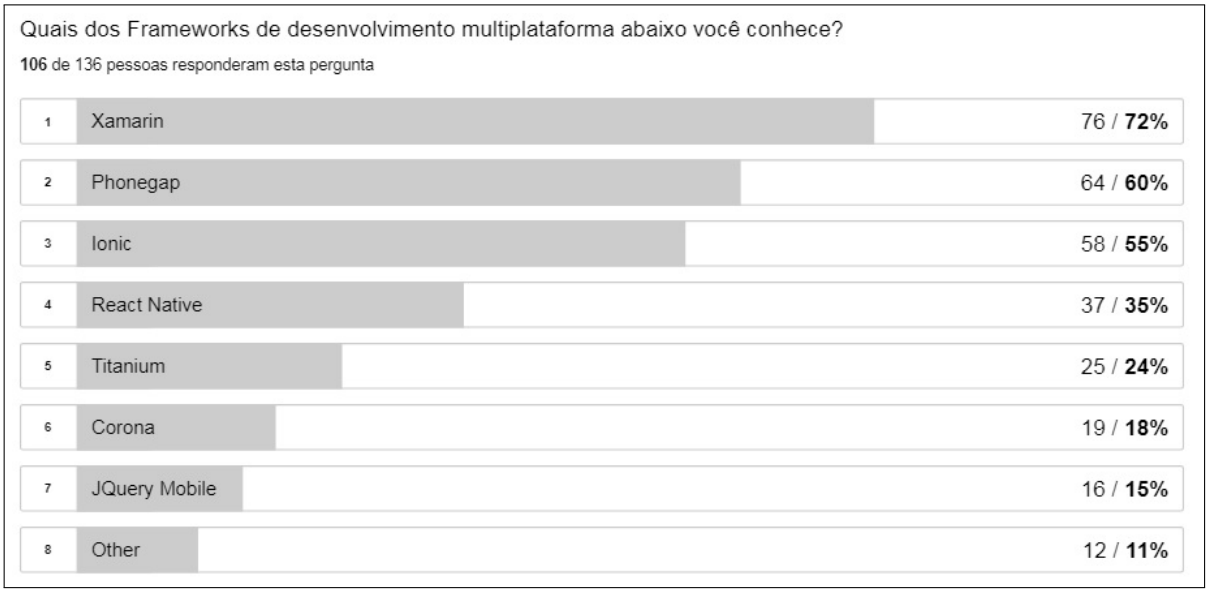
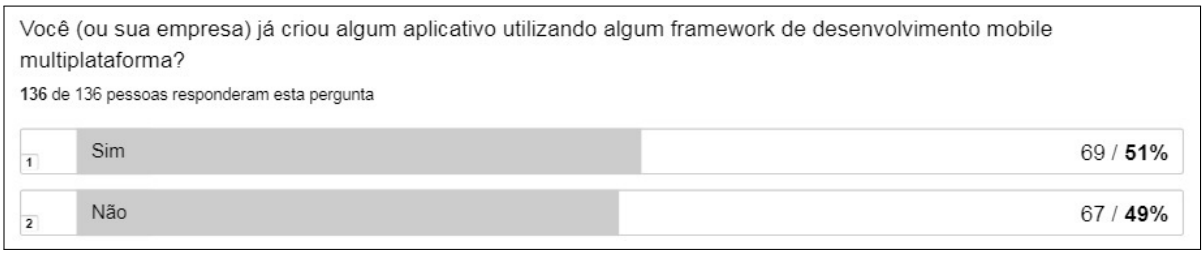


Figura 11 – Uso de *frameworks mobile* multiplataforma.



Na segunda parte desse questionário foi investigada a relevância de alguns fatores na adoção de um *framework mobile* multiplataforma. Na Figura 12 foram dados como opção de resposta 12 recursos presentes em dispositivos que um *framework* precisa suportar. Todos demonstraram ter relevância no processo de decisão sobre o *framework* mais adequado. Alguns se destacaram, como é o caso do suporte a Redes e Dados, GPS e Localização, Armazenamento e Câmera. Esses 4 recursos mostram-se com grande peso para os desenvolvedores que desejam adotar um *framework mobile* multiplataforma. Como mostrado na Figura 13, a importância acerca de 4 itens também foi investigada: tamanho da comunidade, linguagem utilizada pelo *framework*, preço, restrições e licença. Entre os 4 itens, os participantes demonstraram mais preocupação com o tamanho da comunidade e a linguagem de programação utilizada, que juntas somam 75%.

O tamanho da comunidade está diretamente ligado à continuidade do *framework* e ao suporte oferecido na resolução de erros, logo o desenvolvedor estará mais seguro em fazer uso dele. Já a importância que foi dada à linguagem de programação utilizada justifica-se principalmente pela segurança que ela possui e o aproveitamento de conhecimento já adquirido. Apesar disso, 88% dos que responderam ao questionário estão dispostos a aprender outra linguagem, caso necessário, conforme mostra a Figura 16. O preço foi escolhido por 18% dos questionados e ocupou a terceira posição no *ranking*. Esse fato sinaliza que apesar dos desenvolvedores não terem o preço como prioridade máxima, ele ainda é considerado um fator relevante. A Figura 14 mostra que 58% dos entrevistados não pagariam para utilizar um *framework mobile* multiplataforma. Por fim, o suporte à versão mais recente das plataformas, por exemplo, Android, iOS e Windows Phone, demonstrou ser muito importante, ou seja, a grande maioria dos desenvolvedores entende que o *framework* precisa suportar as versões mais recentes das plataformas, isto é, devem evoluir paralelamente.

Figura 12 – Recursos dos dispositivos.

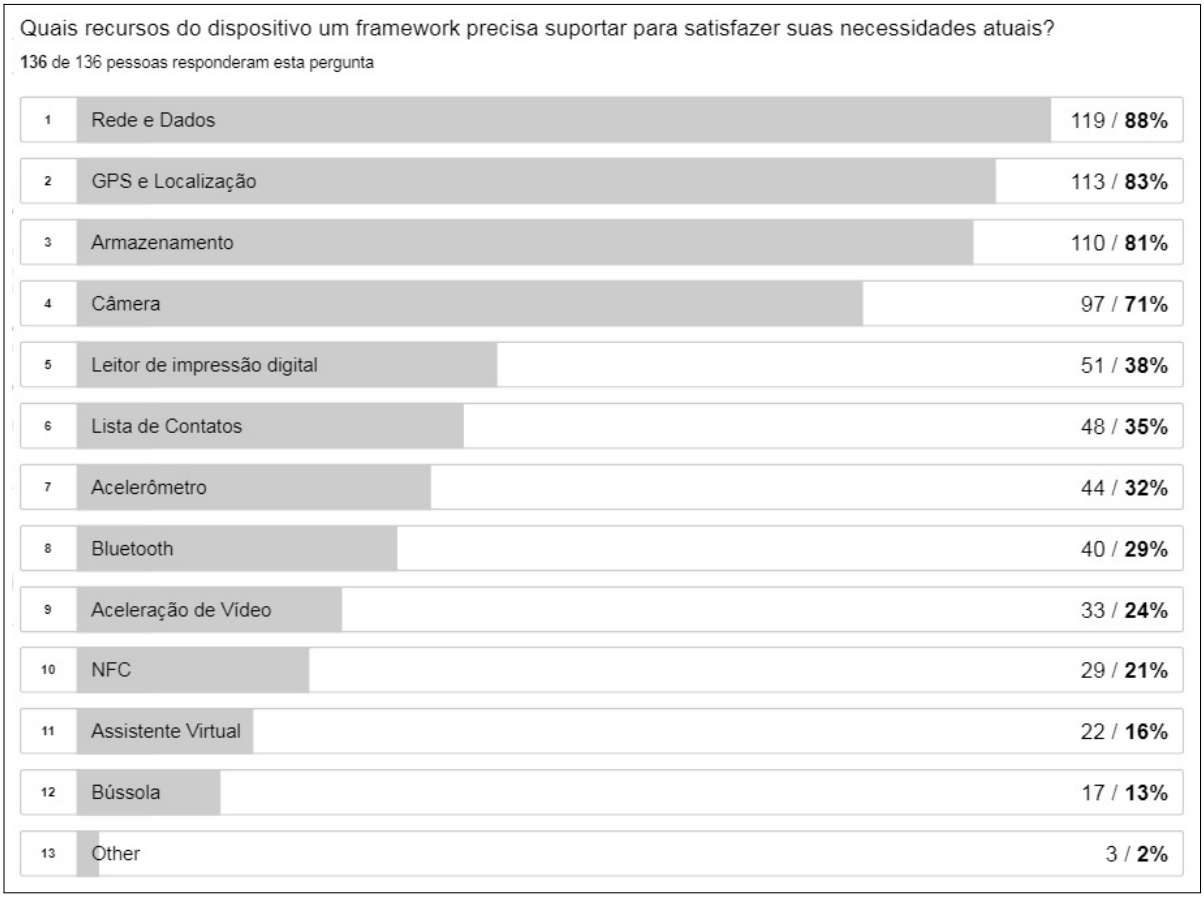


Figura 13 – Nível de relevância dos fatores na escolha de um *framework mobile* multiplataforma.

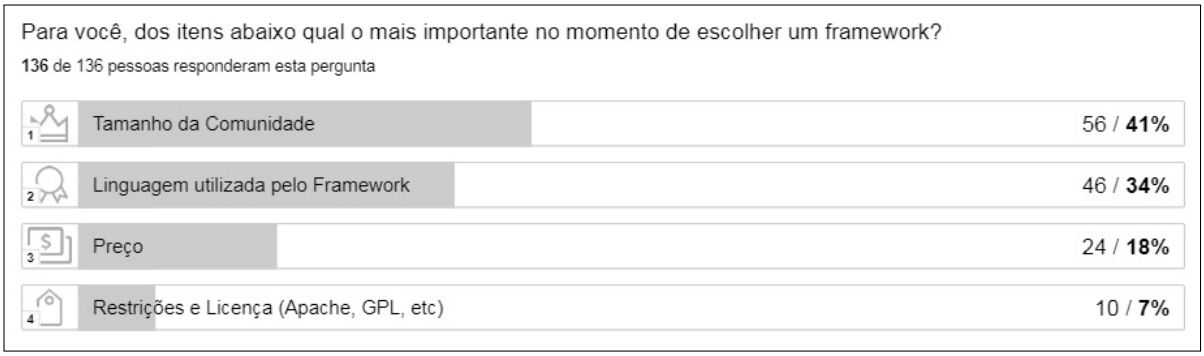


Figura 14 – Desembolso de um valor pelo uso de um *framework mobile* multiplataforma.

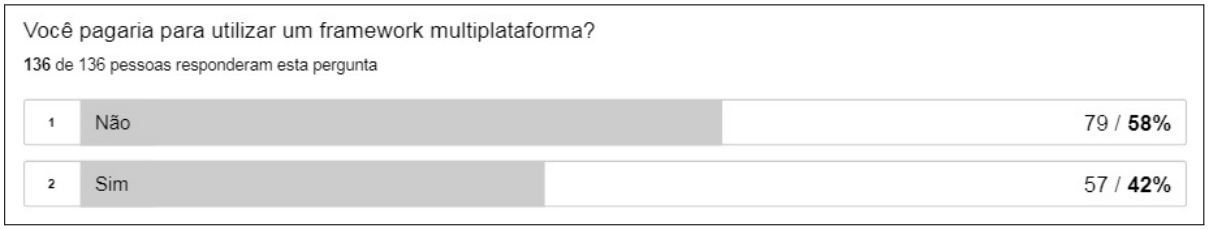


Figura 15 – Suporte à versão mais recente.

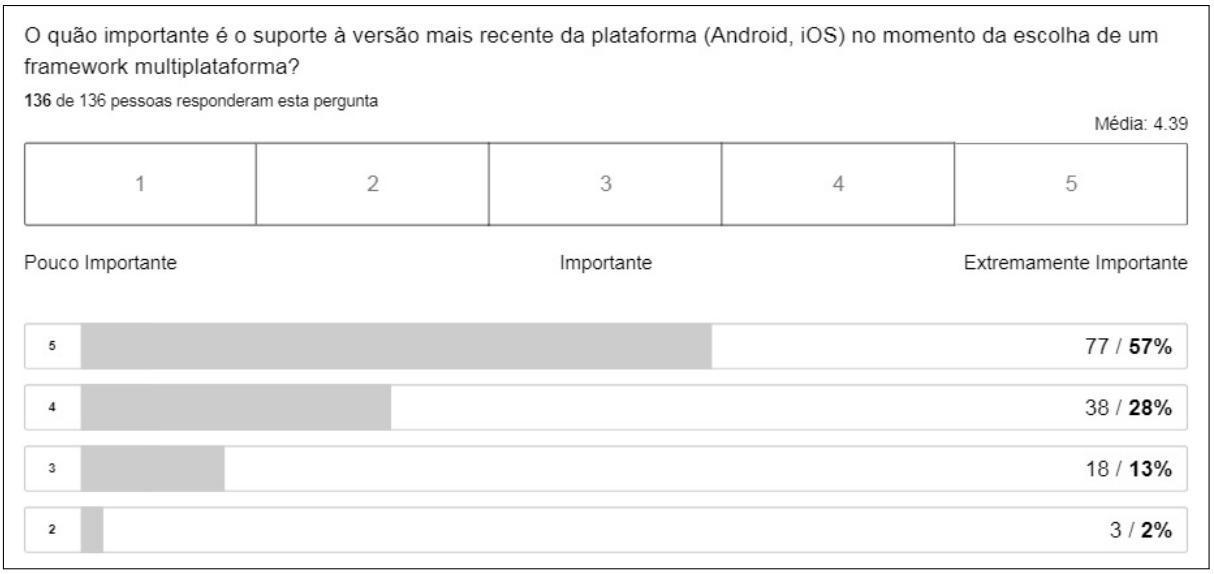
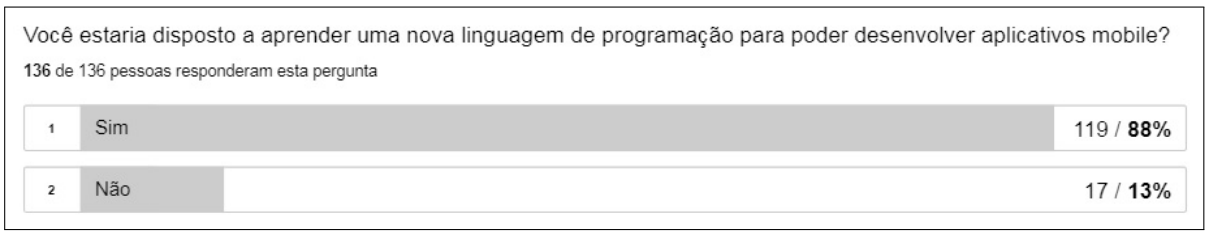


Figura 16 – Aprender nova linguagem de programação.



Como resultado da análise dos dados dessa pesquisa, foi possível definir os critérios de escolha de um *framework* para que possam ser usados na abordagem desenvolvida, a qual é apresentada no capítulo 4. Os critérios são apresentados na próxima seção.

4.6 Definição dos Critérios de Avaliação

Após analisar os dados da pesquisa, considerar a opinião dada em entrevistas informais por desenvolvedores que utilizam *frameworks mobile* multiplataforma e pesquisadores, foi definido um conjunto de doze itens como critérios de adoção de *frameworks mobile* multiplataforma, que são utilizados pelo sistema de recomendação baseado em conhecimento. Abaixo seguem os 12 critérios de escolha, juntamente com a motivação pela qual foram selecionados:

1. **Plataforma suportada**
   Essa característica não poderia deixar de figurar entre as selecionadas, pois o objetivo de um *framework mobile* multiplataforma é justamente disponibilizar aplicativos para mais de uma plataforma. Para reforçar, a Figura 6 mostra que 70% dos participantes da pesquisa já desenvolveram para mais de uma plataforma.

2. **Linguagem de programação utilizada**
   Os *frameworks* suportam diferentes linguagens de programação, e esse pode ser um critério que afaste ou aproxime algum desenvolvedor de algum *framework*. A Figura 16 mostra que, apesar de ser uma pequena porcentagem, existem pessoas que não estão dispostas a aprender uma nova linguagem. Logo, esse é um critério de escolha determinante para elas.

3. **Tipo de aquisição ($)**
   Esse critério é bastante importante, pois a depender do valor que o desenvolvedor tem para financiar o projeto, será possível pagar ou não por um *framework*. A Figura 13 mostra que 24 pessoas também entendem dessa forma. Além disso, a Figura 14 ilustra que, em média, metade dos pesquisados não pagariam para fazer uso de um *framework*, logo, várias opções já seriam dispensadas para esse público.

4. **Licença**
   Esse critério foi apontado por 10 desenvolvedores participantes da pesquisa como importante para ser analisado na escolha de um *framework*. Dessa forma, ficou entendido que para alguns desenvolvedores esse critério é um diferencial.

5. **Suporte a compras internas**
   Muitos aplicativos estão explorando essa característica como forma e monetização. Para que seja possível ao desenvolvedor fazer uso de compras internas, o *framework* precisa fornecer suporte a essa funcionalidade. Portanto, ficou entendido que esse critério é um diferencial entre *frameworks*.

6. **Envia o binário para a loja**
   Notou-se que a tarefa de enviar o binário para a loja se mostra muitas vezes burocrática, e que alguns *frameworks* a abstraem, tornando essa tarefa mais fácil para o desenvolvedor. Dessa forma, ficou entendido que esse critério é um diferencial entre *frameworks*.

7. **Recursos suportados**
   O acesso aos recursos do dispositivo é um critério fundamental, pois essa foi uma grande evolução dos *frameworks*, sendo possível manipular GPS, acelerômetro, entre outros. Os desenvolvedores pesquisados também entendem assim, como ilustrado na Figura 12. Alguns recursos não foram considerados, pois todos os *frameworks* selecionados no capítulo 2 fornecem suporte a eles, logo não seria um diferencial de um em relação a outro.

8. **Geração de *bytecodes***
   A geração de *bytecodes* por um *framework* significa que o código fonte será tanspilado para a linguagem de programação nativa do *framework*, ou seja, é gerado um projeto nativo voltado para a plataforma desejada. Essa pode ser uma característica importante para o desenvolvedor, pois ele poderá seguir desenvolvendo nativamente após a transpilação ou, ainda, questões de desempenho podem ser avaliadas pelo desenvolvedor. Ele pode entender que uma abordagem que gere *bytecodes* tenha um desempenho melhor. Dessa forma, esse é um critério que pode diferenciar um *framework* do outro.

9. **Tempo de adoção de versão mais recente das plataformas suportadas**
   Essa característica foi apontada por 98% dos participantes da pesquisa como um critério importante na escolha de um *framework*, como ilustrado na Figura 15.

10. **Atualizações constantes dos *frameworks***
    A frequência de atualizações pode revelar uma questão importante, como: continuidade do *framework*, dando um sensação ao desenvolvedor que a empresa ou comunidade mantenedora não está abandonando o projeto. Portanto, quando o *framework* fornece atualizações constantes, o desenvolvedor tem uma segurança maior para escolhe-lo.

11. **Suporte a criação de Interface através de editor gráfico**
    Alguns desenvolvedores preferem desenvolver suas tela através de um editor gráfico, não são muito adeptos de codificação na interface, assim o *framework* disponibilizar um editor gráfico é um grande diferencial. Logo, para esse público, um *framework* pode ser mais indicado em relação a outro considerando esse critério.

12. **Utilização de IDE de desenvolvimento**
    Para alguns desenvolvedores que preferem não gastar tempo avaliando e montando ambientes de desenvolvimento, o *framework* disponibilizar uma IDE é um grande diferencial. Dessa forma, para esse público, um *framework* pode levar vantagem em relação a outro considerando esse critério.

O Quadro 4 demonstra os critérios supracitados acompanhados de suas respectivas possíveis respostas.

Quadro 4 – Critérios e Possíveis Respostas.

CRITÉRIOS DE AVALIAÇÃO	POSSÍVEIS VALORES
1. Plataforma suportada	Android, iOS ou Windows Phone
2. Linguagem de programação	React, Javascript + HTML, Typescript, Ruby, Lua e C#
3. Tipo de aquisição ($)	Grátis: Totalmente gratuita, Flexível: Existem opções pagas e gratuitas, Pago: Totalmente pago
4. Licença	*Open Source - MIT License, Open Source - Apache 2 License*, Proprietária
5. Possui suporte a compras internas	Indiferente ou Sim
6. Envia o binário para as lojas	Indiferente ou Sim
7. Recursos suportados	Bússola, GPS, Acelerômetro, 3D *Touch*, Assistente Virtual, NFC, Leitor de Impressão Digital ou Aceleração de Vídeo
8. Geração de *Bytecodes*	Indiferente, Não ou Sim
9. Tempo de adoção de versão mais recente das plataformas suportadas	Imediato, Até 3 meses, 3 a 6 meses ou Acima de 6 meses
10. Atualizações constantes do *framework*	Indiferente ou Sim
11. Suporte a criação de interface	Indiferente ou Sim
12. Utilização de IDE de desenvolvimento	Indiferente ou Sim

Todas as opções de respostas aos critérios foram definidas considerando os principais *frameworks* definidos no capítulo 2. Por exemplo, para o critério número 2, que trata acerca das linguagens de programação suportadas pelo *framework*, são disponibilizadas 6 opções de resposta (React, Javascript e HTML, Typescript, Ruby, Lua e C#), pois cada linguagem citada compõe um conjunto de linguagens suportadas pelos *frameworks* selecionados.

4.7 Considerações Finais do Capítulo

Neste capítulo foram identificados os principais critérios que devem ser considerados na abordagem que visa possibilitar a escolha do *framework mobile* multiplataforma mais indicado para um determinado desenvolvedor. A identificação foi realizada após a aplicação e análise de um *survey* que foi distribuído em listas de discussão compostas por desenvolvedores *mobile*.

O próximo capítulo apresentará uma abordagem que, considerando os critérios de escolha identificados neste capítulo e os principais *frameworks* selecionados no capítulo 2, será capaz de indicar qual *framework mobile* multiplataforma será o mais indicado para um determinado desenvolvedor.

5
Abordagem de Recomendação

No capítulo anterior discutiu-se acerca dos fatores que determinam a escolha do *framework* mais indicado a determinado desenvolvedor. A partir desse fatores tornou-se possível determinar a similaridade entre os desejos do desenvolvedor e os 8 *frameworks* selecionados, e assim produzir um *ranking* de recomendação, onde o primeiro colocado é o mais indicado ao desenvolvedor e o último o menos indicado.

5.1 Abordagem

A abordagem de recomendação baseada em conhecimento se encaixa perfeitamente ao cenário estudado, pois ela faz a recomendação em função de uma base de conhecimento, que neste trabalho é composta pelas características dos *frameworks* (recursos suportados, tipo de licença, etc), definidas no capítulo 4. Para compor essa base, foi feita uma pesquisa no site oficial de cada *framework*, e em seguida as informações foram persistidas no banco de dados. Com essas informações persistidas, foi possível compará-las com os critérios desejados pelo desenvolvedor.

Para conhecer os desejos do desenvolvedor, ou seja, o que ele necessita que componha o *framework* desejado por ele, foi montado um questionário que aborda em suas perguntas cada critério definido no capitulo 4. A seguir estão as perguntas e suas possíveis respostas:

1. Deseja que quais plataformas sejam suportadas pelo *framework*? **(Admite várias opções simultâneas)**

    - Android
    - iOS
    - Windows Phone

2. Quais linguagens você estaria disposto a usar para desenvolver seu app? **(Admite várias opções simultâneas)**

    - React
    - Javascript + HTML
    - TypeScrip
    - Ruby
    - Lua
    - C#

3. Qual o tipo de aquisição você deseja utilizar para ter o *framework* desejado? **(Admite apenas uma opção)**

    - Grátis - Totalmente gratuita
    - Flexível - Existem opções pagas e gratuitas
    - Pago - Totalmente pago

4. Qual o tipo de licença você deseja utilizar? **(Admite apenas uma opção)**

    - *Open Source - MIT License*
    - *Open Source - Apache 2 License*
    - Proprietária

5. Deseja que o *framework* suporte o desenvolvimento de apps com compras internas? **(Admite apenas uma opção)**

    - Indiferente
    - Sim

6. Deseja que o *framework* possua a possibilidade de enviar o app desenvolvido para a loja? **(Admite apenas uma opção)**

    - Indiferente
    - Sim

7. Quais recursos que você deseja que o *framework* suporte? **(Admite várias opções simultâneas)**

    - Bússola
    - GPS

- Acelerômetro
- *3D Touch*
- Assistente Virtual
- NFC
- Leitor de Impressão Digital
- Aceleração de Vídeo

8. Deseja que o framework gere os *bytecodes* específicos de cada plataforma? **(Admite apenas uma opção)**

   - Indiferente
   - Não
   - Sim

9. Quanto tempo você deseja esperar até que o *framework* suporte a versão mais recente das plataformas suportadas (iOS, Android, entre outros)? **(Admite apenas uma opção)**

   - Imediato
   - Até 3 meses
   - De 3 a 6 meses
   - Acima de 6 meses

10. Deseja que o *framework* tenha atualizações constantes (correções de *bugs*, novas funcionalidades)? **(Admite apenas uma opção)**

    - Indiferente
    - Sim

11. Deseja que o *framework* forneça um editor gráfico para a criação de interfaces? **(Admite apenas uma opção)**

    - Indiferente
    - Sim

12. Deseja que o *framework* forneça uma IDE própria de desenvolvimento? **(Admite apenas uma opção)**

    - Indiferente
    - Sim

As perguntas acimas demandam estilos de respostas diferentes. Por exemplo, a questão 1 fornece três opções para resposta, que permitem que sejam escolhidas simultaneamente. A questão 3 também fornece três opções, mas só permite a escolha de um item como resposta. Como existe essa diferença entre as questões, diferentes medidas de similaridades são necessárias. Após analisar as questões, ficaram estabelecidos 3 grupos de questões:

- Grupo 1: Questão 8.
- Grupo 2: Questões 1, 2, 5, 6, 7, 10, 11 e 12.
- Grupo 3: Questões 3, 4 e 9.

A formação dos grupos foi definida segundo as medidas de similaridade que se encaixaram com cada questão. Para a questão do grupo 1 foi utilizada a medida de Jaccard; Para as questões do grupo 2 também foi utilizada a medida de Jaccard, mas com uma modificação na equação; e para as questões do grupo 3 foi usada a medida Euclidiana.

- Grupo 1:

  Como explicado no capítulo 2, seção 3.1, a medida de Jaccard é dada pela equação abaixo:

  $$dist(X_i, Y_i) = 1 - \frac{a}{a+b+c}$$

  onde, considerando a questão 8 como exemplo:

  a = 1 quando o usuário deseja que o *framework* gere *bytecodes* e que essa característica esteja presente no *framework* comparado. Caso contrário "a" receberá o valor 0.

  b = 1 quando o usuário deseja que o *framework* gere *bytecodes* e que essa característica não esteja presente no *framework* comparado. Caso contrário "b" receberá o valor 0.

  c = 1 quando o usuário não deseja que o *framework* gere *bytecodes* e que essa característica esteja presente no *framework* comparado. Caso contrário "c" receberá o valor 0.

  Então, quanto mais situações semelhantes a "a", mais similares eles são. Por outro lado, quanto mais situações semelhantes a "b"e "c", mais dissimilares eles são. Porém, quando o usuário escolhe a opção "Indiferente", nenhum cálculo é realizado, ou seja, esse fator não interfere na escolha do *framework* para ele.

- Grupo 2:

  Para esse grupo também será utilizada a medida de Jaccard para cada questão, porém com uma alteração em sua equação. A condição representada pela letra "c" não será

considerada, pois caso o usuário não deseje uma determinada característica e o *framework* a tenha, este não se adequará nem mais nem menos aos desejos daquele. A equação foi definida como a seguir:

$$dist(X_i, Y_i) = 1 - \frac{a}{a+b}$$

onde, considerando a questão 1 como exemplo:

a = número de ocorrências onde o usuário deseja que dada plataforma seja suportada e essa característica esteja presente no *framework* comparado. Por exemplo, caso o usuário deseje que o *framework* suporte Android e iOS, quando a comparação é feita com o Corona, que suporta ambos, "a" receberia o valor 2.

b = número de ocorrências onde o usuário deseja que dada plataforma seja suportada e essa característica não esteja presente no *framework* comparado. Por exemplo, caso o usuário deseje que o *framework* suporte Windows Phone, quando a comparação é feita com o Titanium, que não possui suporte a ele, "b" receberia o valor 1.

c = número de ocorrências onde o usuário não deseja dada plataforma seja suportada e essa característica esteja presente no *framework* comparado. Por exemplo, caso o usuário não deseje que o *framework* suporte Windows Phone, quando a comparação é feita com o Xamarin, que suporta esta plataforma, "c" receberia o valor 1. Mas como destaca acima, essa condição não será utilizada.

Então, quanto mais situações semelhantes a "a", mais similares eles são. Por outro lado, quanto mais situações semelhantes a "b", mais dissimilares eles são.

- Grupo 3:

Esse grupo é formado pelas questões 3, 4 e 9. Essas questões apresentam suas possibilidades de resposta como enumerados, logo só permitem uma única opção de resposta.

  - Questão 3 (Qual o tipo de aquisição você deseja utilizar para ter o framework desejado?)

    As opções de resposta para essa questão são: Grátis - Totalmente gratuita $\rightarrow$ 0; Flexível - Existem opções pagas e gratuitas $\rightarrow$ 0,5; Pago - Totalmente pago $\rightarrow$ 1. Para cada opção de resposta é atribuído um valor (transparente para o usuário), onde a mínima distância seja zero e a máxima seja 1, assim se estabelece uma normalização com as demais medidas. Considerando que "a" é o valor escolhido pelo usuário e "b" a característica do *framework* comparado, a distância pode ser calculada pela equação a seguir:

$$dist(X_i, Y_i) = |a - b|$$

- Questão 4 (Qual o tipo de licença você deseja utilizar?)

  As opções de resposta para essa questão são: *Open Source - MIT License* $\rightarrow$ 0; *Open Source - Apache 2 License* $\rightarrow$ 0; Proprietária $\rightarrow$ 1. Para cada opção de resposta é atribuído um valor (transparente para o usuário), onde a mínima distância seja zero e a máxima seja 1, assim se estabelece uma normalização com as demais medidas. As duas opções *Open Source* possuem o mesmo valor (zero), pois não existe hierarquia do Apache 2 sobre o MIT e vice-versa. Considerando que "a" é o valor escolhido pelo usuário e "b" a característica do *framework* comparado, a distância pode ser calculada pela equação a seguir:

$$dist(X_i, Y_i) = |a - b|$$

- Questão 9 (Quanto tempo você deseja esperar até que o framework suporte a versão mais recente das plataformas suportadas (iOS, Android, entre outros)?)

  As opções de resposta para essa questão são: Imediato $\rightarrow$ 0; Até 3 meses $\rightarrow$ 1/3; De 3 a 6 meses $\rightarrow$ 2/3; Acima de 6 meses $\rightarrow$ 3/3. Para cada opção de resposta é atribuído um valor (transparente para o usuário), onde a mínima distância seja zero e a máxima seja 1, assim se estabelece uma normalização com as demais medidas.

  Essa questão tem uma particularidade. Caso o usuário escolha uma opção menos restritiva que a característica do *framework* comparado a distância será zero. Por exemplo, supondo que o usuário diz que a espera pelo suporte pode ser acima de 6 meses e o *framework* comparado tenha suporte imediato às versões mais recentes das plataformas suportadas, a distância entre esse fatores é zero, pois se o usuário está disposto a esperar mais de 6 meses, o imediato está contido nessa opção.

  Quando não houver a particularidade supracitada, o cálculo será realizado como segue: considerando que "a" é o valor escolhido pelo usuário e "b" a característica do *framework* comparado, a distância pode ser calculada pela equação a seguir:

$$dist(X_i, Y_i) = |a - b|$$

Após realizar o cálculo para cada critério abordado nas questões, seus valores foram utilizados para o calculo final da similaridade, dado pela seguinte equação:

$$Sim(X, Y) = exp(-D(X, Y))$$

onde X é o vetor de desejos do usuário, abordado nas questões, e Y é o vetor de atributos do *framework* comparado. E $D(X, Y)$ é calculado através da distância euclidiana ponderada:

$$D(X, Y) = (\frac{1}{\sum iw_i})^{\frac{1}{2}} * [\sum iw_i * dist(X_i, Y_i)^2]^{\frac{1}{2}}$$

onde 0 <= w_i <= 1 (peso), i = 1..n (número de atributos), $dist(X_i, Y_i)$ é a métrica de similaridade específica para cada tipo de atributo, como foi definido acima para nos grupos 1, 2 e 3.

A euclidiana ponderada é utilizada pois para cada fator indicado pelo usuário, ele também deve informar o nível de importância deste fator para ele. O usuário deve escolher um valor entre 1 (pouquíssimo relevante) e 5 (muitíssimo relevante). Porém, este valor deve ser normalizado para que a soma deles resulte 1. A normalização foi calculada dividindo cada um dos pesos pela soma de todos os pesos escolhidos. Por exemplo, supondo que o usuário forneceu a relevância de cada questão: 1, 3, 3, 2, 3, 4, 5, 5, 5, 5, 5, 1, a soma total das relevâncias daria 42. Então cada peso é dividido por 42, assim, seus pesos normalizados ficam: 0.023, 0.071, 0.071, 0.047, 0.071, 0.095, 0.119, 0.119, 0.119, 0.119, 0.119 e 0.023, com soma igual a 1.

O algoritmo 1 ilustra o mecanismo da abordagem de recomendação de *frameworks mobile* multiplataforma definida.

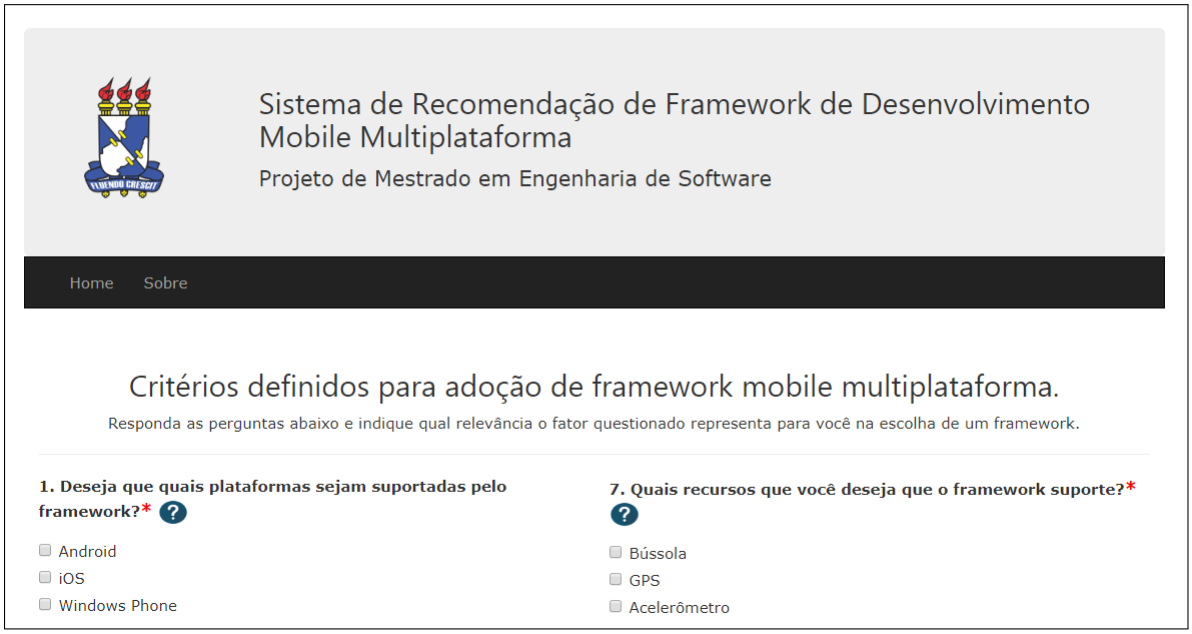
Algoritmo 1: *Ranking*

```
   Entrada: respostasUsuarios
   Saída: ranking[]
 1 início
     // Consulta ao banco de dados.
 2   frameworks ← obterFrameworks();
 3   j ← 0
 4   pontuacaoFramework ← null;
 5   for framework : frameworks do
       // Consulta ao banco de dados.
 6     caracteristicas[] ← obterCaracteristicasFramework(framework);
 7     i ← 0;
 8     pontuacao ← 0;
 9     pontuacaoFramework ← new PontuacaoFramework(framework.nome);
10     for respostaUsuario : respostasUsuarios do
         // A similaridade é obtida considerando a medida adequada
            para cada tipo de resposta/característica.
11       pontuacaoQuestao ← respostaUsuario.peso *
          obterSimilaridade(caracteristicas[i], respostaUsuario);
12       i ← i + 1;
13       pontuacao ← pontuacao + pontuacaoQuestao;
14     end
15     pontuacaoFramework.pontuacaoQuestao ← pontuacao;
16     ranking[j] ← pontuacaoFramework;
17     j ← j + 1;
18     caracteristicas[] ← null;
19   end
20   sort(ranking[]);
21   return ranking[]
22 fim
```

5.2 Aplicação Web

Foi criado um sistema de recomendação de *frameworks mobile* para desenvolvimento multiplataformas utilizando a abordagem exposta acima, que envolveu a definição de dois fatores: a seleção dos principais *frameworks* e o levantamento dos fatores que determinam a escolha do *framework* mais indicado a um dado desenvolvedor. A Figura 17 mostra uma visão geral da interface do sistema.

Figura 17 – Visão Geral.



A aplicação se apresenta no formato de questionário eletrônico, onde o usuário é incentivado a responder as perguntas, que serão usadas como parâmetro para que a recomendação aconteça, conforme ilustra a Figura 18. Para cada pergunta será necessário que o usuário defina o nível de importância que a pergunta tem para ele. Caso o usuário tenha alguma dúvida sobre o que foi perguntado, ele poderá clicar no ícone representado por um interrogação, assim uma janela será apresentada com a devida explicação sobre a pergunta, como é exposto na Figura 19.

Figura 18 – Pergunta.

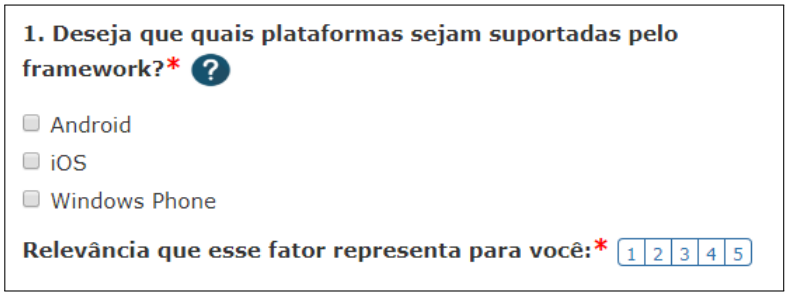
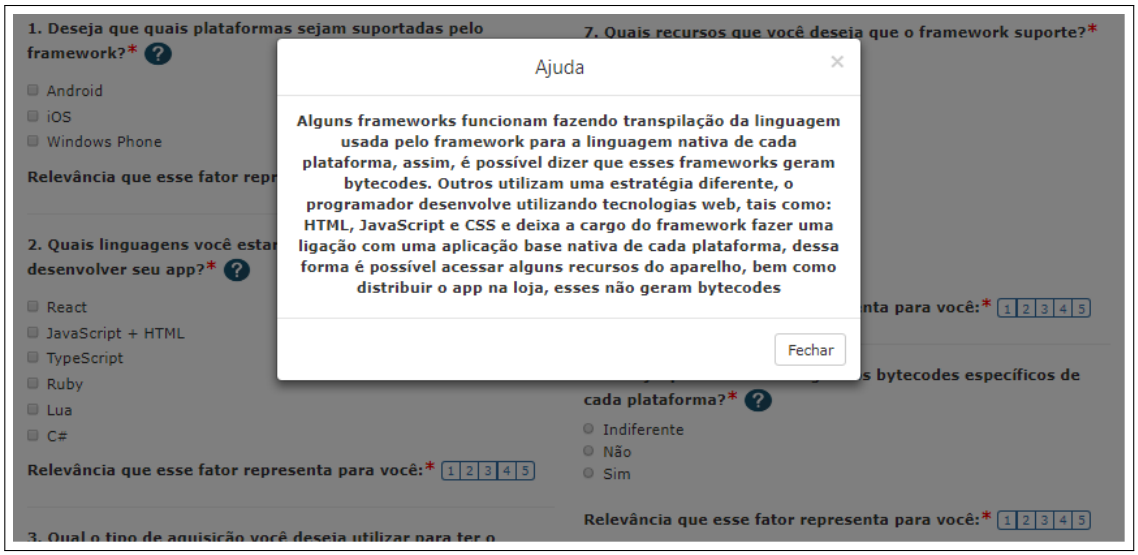
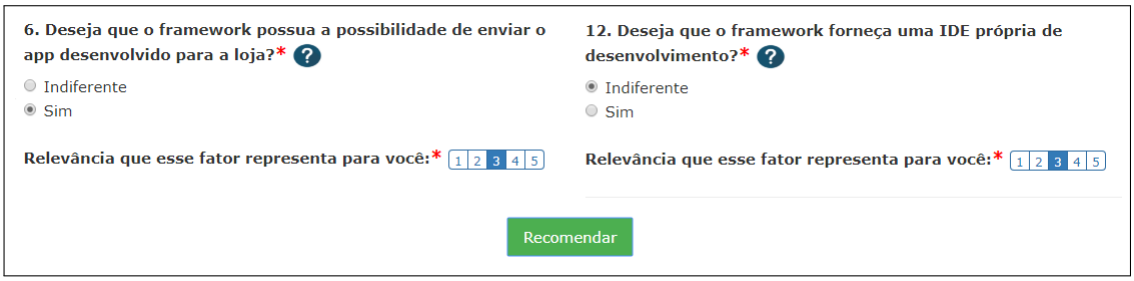


Figura 19 – Ajuda.



Após o desenvolvedor responder todas as perguntas e atribuir seus respectivos níveis de relevância, ou seja, após ele deixar claro o que deseja em um *framework mobile* multiplataforma, o botão recomendar deve ser pressionado, conforme mostrado na Figura 20.

Figura 20 – Botão Recomendar.



A recomendação é exibida no formato de um *ranking* em ordem decrescente de importância, onde o primeiro colocado, exibido com uma medalha de ouro ao seu lado esquerdo, é o mais indicado ao usuário, e o último o menos indicado. O *ranking* pode ser visto na Figura 21.

Caso o usuário deseje saber um pouco mais sobre os *frameworks* que compõem o *ranking*, ele deverá clicar sobre o nome do *framework* para que um breve resumo sobre cada um deles, acompanhado de suas características, seja apresentado. A Figura 22 mostra o detalhe do React Native.

A tabela exposta na Figura 23 apresenta expressões faciais e cores para representar quanto uma característica assinalada pelo usuário corresponde a um característica presente no *framework*. Essa tabela serve para que o usuário tenha mais transparência do processo de recomendação. Na parte de baixo da Figura 23 pode ser vista uma legenda que explica o significado de cada expressão. A carinha vermelha representa 0% de correspondência entre o fator analisado e o desejo do usuário, a laranja 25%, a amarela 50%, a verde claro 75% e por fim, a verde escuro 100%. Existe também um traço preto que é mostrado quando o usuário marca a questão como

Figura 21 – Ranking dos *frameworks*.

Frameworks mobile multiplataforma mais adequado as suas prioridades.

Os frameworks abaixo são apresentados por ordem de adequação as suas necessidades.

- **React Native**
- **Phonegap**
- **Corona**
- **Ionic**
- **Xamarin**
- **Appcelerator Titanium**
- **Sencha**
- **Rhodes**

Figura 22 – Detalhe do *Framework*.

React Native

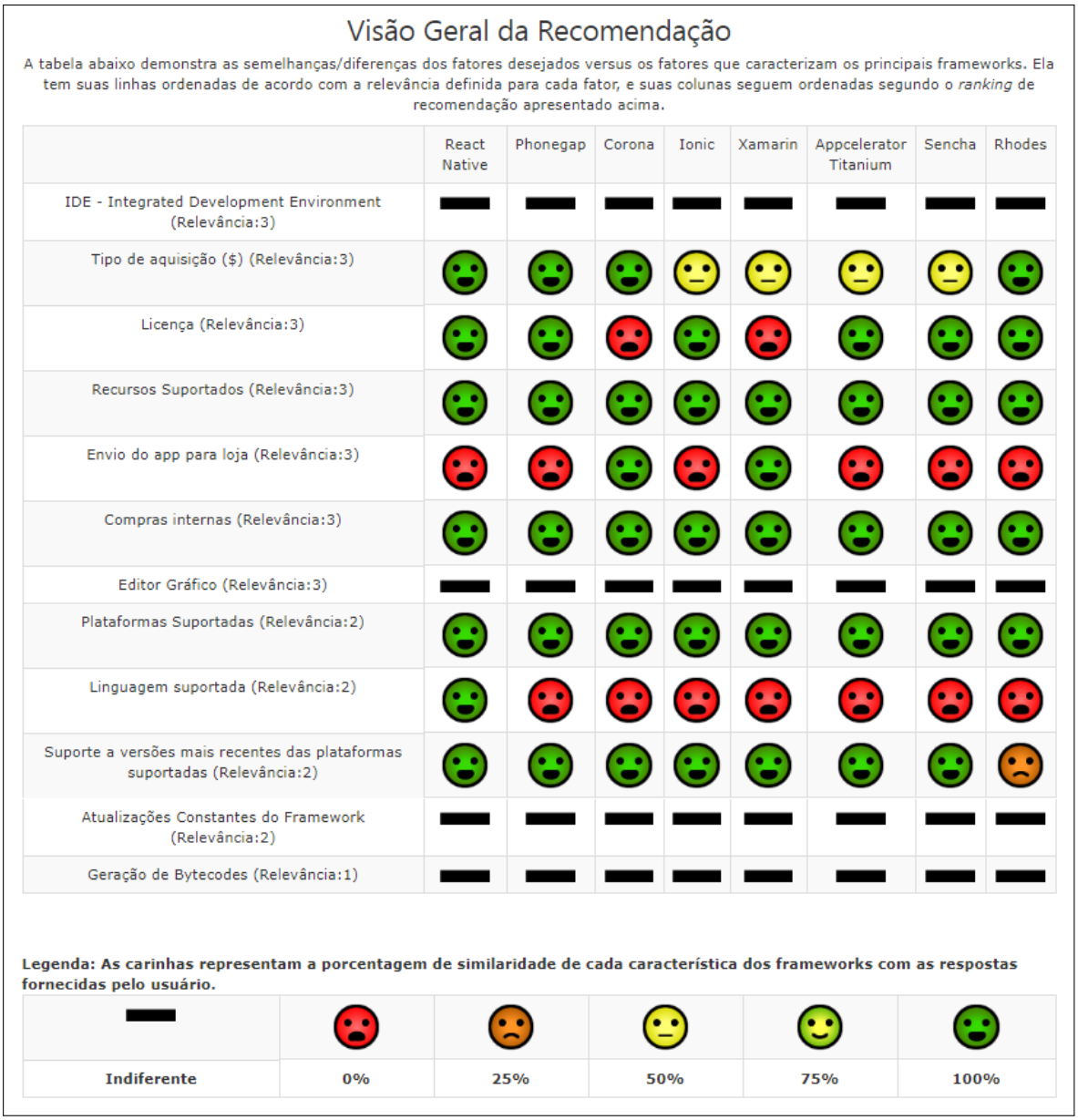
Com o React Native, é possível criar, de forma gratuita, a partir de um único código programado em javascript com a biblioteca react, um aplicativo móvel multiplataforma, que segundo o Facebook, criador e mantenedor do framework, é igual a um aplicativo criado usando o Objective-C ou o Java. React Native combina componentes escritos em Objective-C, Java ou Swift, assim, é simples descer para o código nativo se você precisar otimizar alguns aspectos do seu aplicativo. Também é fácil criar uma parte do seu aplicativo no React Native e outra usando código nativo diretamente - é assim que funciona o aplicativo do Facebook.
Saiba Mais: **https://facebook.github.io/react-native/**

Plataformas Suportadas: iOS, Android e Windows Phone
Linguagem suportada: Javascript + React
Tipo de aquisição ($): Grátis
Licença: Open-Source (MIT License)
Compras internas: Sim
Envio do app para loja: Não
Recursos Suportados: Bússola, GPS, Acelerômetro, 3D Touch, Assistente Virtual, NFC, Leitor de Impressão Digital, Aceleração de Vídeo
Geração de Bytecodes: Não
Suporte a versões mais recentes das plataformas suportadas: Imediato
Atualizações Constantes do Framework: Sim
Editor Gráfico: Não
IDE - Integrated Development Environment: Não

- **Phonegap**
- **Corona**
- **Ionic**
- **Xamarin**
- **Appcelerator Titanium**
- **Sencha**
- **Rhodes**

indiferente para ele. A tabela ordena suas linha de acordo com a relevância de cada questão para o usuário, as mais relevantes aparecem nas primeiras posições e as menos relevantes mais abaixo. Sua colunas são ordenadas de acordo com o *ranking* apresentado na Figura 21, ou seja, o *framework* mais à esquerda é o mais indicado e o mais à direita o menos indicado.

Figura 23 – Tabela Explicativa.



Visão Geral da Recomendação

A tabela abaixo demonstra as semelhanças/diferenças dos fatores desejados versus os fatores que caracterizam os principais frameworks. Ela tem suas linhas ordenadas de acordo com a relevância definida para cada fator, e suas colunas seguem ordenadas segundo o *ranking* de recomendação apresentado acima.

	React Native	Phonegap	Corona	Ionic	Xamarin	Appcelerator Titanium	Sencha	Rhodes
IDE - Integrated Development Environment (Relevância:3)	—	—	—	—	—	—	—	—
Tipo de aquisição ($) (Relevância:3)	100%	100%	100%	50%	50%	50%	50%	100%
Licença (Relevância:3)	100%	100%	0%	100%	0%	100%	100%	100%
Recursos Suportados (Relevância:3)	100%	100%	100%	100%	100%	100%	100%	100%
Envio do app para loja (Relevância:3)	0%	0%	100%	0%	100%	0%	0%	0%
Compras internas (Relevância:3)	100%	100%	100%	100%	100%	100%	100%	100%
Editor Gráfico (Relevância:3)	—	—	—	—	—	—	—	—
Plataformas Suportadas (Relevância:2)	100%	100%	100%	100%	100%	100%	100%	100%
Linguagem suportada (Relevância:2)	100%	0%	0%	0%	0%	0%	0%	0%
Suporte a versões mais recentes das plataformas suportadas (Relevância:2)	100%	100%	100%	100%	100%	100%	100%	25%
Atualizações Constantes do Framework (Relevância:2)	—	—	—	—	—	—	—	—
Geração de Bytecodes (Relevância:1)	—	—	—	—	—	—	—	—

Legenda: As carinhas representam a porcentagem de similaridade de cada característica dos frameworks com as respostas fornecidas pelo usuário.

—	0%	25%	50%	75%	100%
Indiferente	**0%**	**25%**	**50%**	**75%**	**100%**

Por fim, no menu superior, na opção Sobre, como pode ser visto na Figura 24, uma breve descrição sobre o projeto e a lista de *frameworks mobile* multiplataforma considerada nessa aplicação é apresentada. O intuito dessa opção é explicar os objetivos do projeto com a finalidade de contextualizar o usuário do sistema acerca do que se espera.

Figura 24 – Descrição do sistema.

O sistema de recomendação desenvolvido é escalável quanto a atualização e adição de novos critérios e *frameworks*. Para adicionar novos critérios, é preciso atualizar a tela que questiona quais são os desejos do desenvolvedor e definir sua medida de similaridade mais adequada. Quanto à adição de novos *frameworks* é necessário pesquisar suas características e cadastrá-las na base de dados. Dessa forma, o algoritmo passará a considerar tanto a nova característica no processo de similaridade, quanto o novo *framework* passará a ser considerado no *ranking* final.

5.2.1 Desenvolvimento

O sistema de recomendação foi codificado do início ao fim sem fazer uso de bibliotecas de recomendação. Todo o desenvolvimento foi conduzido utilizando a linguagem de programação Python[1] e o banco de dados MySql[2]. A seguir são apresentados todo o ferramental utilizado:

- Python: linguagem de programação utlizada para o desenvolvimento do *back-end*. A versão utilizada foi a 3.6.
- Flask[3]: *framework* utilizado para prover a comunicação feita via *webservice* entre *front-end*

[1] https://www.python.org/
[2] https://www.mysql.com/
[3] http://flask.pocoo.org/

e *back-end*.

- Pythonanywhere[4]: maquina virtual disponibilizada na nuvem com ambiente Python e o framework Flask instalados. Esse ambiente mostrou-se viável por abstrair toda a infraestrutura do desenvolvedor.
- AngularJS[5]: *framework* mantido pelo Google que abstrai as complexidades do javascript no desenvolvimento do *front-end*.
- Bootstrap[6]: *framework front-end* responsivo utilizado na aplicação para trazer mais conforto visual à interface.
- Json[7]: formato de dados que a aplicação utilizou para trafegar os dados entre *front-end* e *back-end*.
- Bitbucket[8]: repositósito onde o código foi versionado.
- Inkscape[9]: ferramenta onde foram trabalhadas as figuras presentes na aplicação.
- Selenium[10]: ferramenta de teste usada para testar a interface da aplicação.
- MySQL: banco de dados utilizado para persistir as informações dos frameworks. Uma instância do MySQL é disponibilizada no Pythonanywhere.

O sistema foi desenvolvido utilizando o padrão MVC. Na parte da *view* foi construída a página, que utilizou: HTML, AngularJS e o Bootstrap. No *model* é onde está toda a regra de negócio (*backend*), ou seja, o algorítmo responsável por fazer os cálculos de similaridade e montar o *ranking*, assim como fazer acesso a base de dados (MySql). O *backend* foi todo desenvolvido utilizando a linguagem de programação Python. Os atributos de todos os *frameworks*, persistidos no banco de dados, foram comparados, utilizando as devidas medidas de similaridades, com os desejos do usuário. O *controller* foi o responsável por fazer a comunicação da *view* com o *model*. Essa comunicação é feita transferindo dados de uma camada para outra, utilizando o Json através do Flask. Toda a infraestrutura foi abstraída na nuvem através da plataforma Pythonanywhere.

5.3 Considerações Finais do Capítulo

Neste capítulo foi apresentada a abordagem de recomendação baseada em conhecimento, a qual fez uso de medidas de similaridade e dos critérios de escolha identificados no capítulo

[4] https://www.pythonanywhere.com/
[5] https://angularjs.org/
[6] https://getbootstrap.com/
[7] https://www.json.org/json-pt.html
[8] https://bitbucket.org/
[9] https://inkscape.org/pt-br/
[10] https://www.seleniumhq.org/

4. Seu objetivo foi descobrir, dentro do conjunto dos principais *frameworks* identificados no capítulo 2, qual deles é o mais similar aos desejos de um dado desenvolvedor.

O próximo capítulo apresentará uma análise que teve por objetivo validar a efetividade da abordagem discutida neste capítulo junto à comunidade desenvolvedora de aplicativos móveis.

6

Avaliação da Abordagem de Recomendação

Este capítulo discute sobre a validação da abordagem desenvolvida e apresentada no capítulo anterior. Para isso foi criado um questionário de pesquisa *online* com o objetivo de investigar a efetividade da solução diante de um público especialista em desenvolvimento móvel. Além disso, a efetividade da lista de *frameworks mobile* multiplataforma selecionados também foi pesquisada, a fim de confirmá-la frente aos especialistas. Por fim, os fatores relevantes na escolha de um *framework mobile* multiplataforma também foram explorados, com o objetivo de confirmá-los como sendo, de fato, relevantes.

6.1 Objetivos Específicos

O questionário foi criado em meio eletrônico através da ferramenta *Google Forms*[1], com o objetivo de avaliar a abordagem desenvolvida neste trabalho. Foram definidos 3 objetivos específicos:

- Avaliar a efetividade da solução diante de um público especialista em desenvolvimento móvel.
- Avaliar se os fatores abordados na pesquisa discutidos no capítulo 4 são realmente relevantes para a escolha de um *frameworks mobile* multiplataforma.
- Avaliar se a lista dos 8 *frameworks mobile* multiplataforma pesquisados são os mais relevantes na visão dos desenvolvedores *mobile*.

[1] <https://www.google.com/forms/about/>

6.2 Projeção do Questionário

O questionário foi elaborado com questões que se dividem em dois grupos: informações pessoais e o conhecimento em desenvolvimento móvel dos participantes, e informações sobre a efetividade da abordagem desenvolvida, fatores de relevância de escolha de um *framework mobile* multiplataforma e sobre a lista dos *frameworks* utilizada.

O primeiro grupo de perguntas aborda as informações pessoais e o conhecimento em desenvolvimento móvel do participante, a fim de conhecer um pouco melhor sobre quem está respondendo as questões. É preciso certificar-se de que os perguntados vivenciam ambientes que lidam com este tipo de desenvolvimento, pois a relevância dessa pesquisa depende da expertise dos perguntados. Os participantes responderam aos seguintes questionamentos:

1. Qual a sua idade?
2. Qual o seu gênero?
3. Qual a sua situação profissional?
4. Qual a sua escolaridade?
5. Qual o seu grau de conhecimento em desenvolvimento *mobile*?

O nome e o e-mail também foram perguntados, mas de forma opcional. Quando essas perguntas foram respondidas, o resultado da pesquisa foi enviado para elas.

O segundo grupo de questões tem o objetivo de avaliar a efetividade da abordagem desenvolvida diante de um público experiente na área de desenvolvimento móvel. Seguem as perguntas deste grupo:

1. O sistema considera 8 frameworks (React Native, Ionic, Phonegap, Corona, Xamarin, Titanium, Sencha e Rhodes) para fazer a recomendação, você considera uma lista relevante? Adicionaria algum framework à lista acima? Quais?
2. Você considera relevantes os critérios abordados nas perguntas para a escolha de um framework? Adicionaria algum critério? Quais?
3. Como você avalia o ranking de frameworks recomendado? Qual o seu nível de concordância com o resultado? Alguma observação?
4. Você considera essa ferramenta relevante para a escolha de um framework mobile múltiplas plataformas?
5. Você recomendaria essa ferramenta para um desenvolvedor ou empresa? Qual seria essa probabilidade? Alguma observação?

6. Como você avalia a usabilidade da ferramenta? Alguma observação?

7. De forma geral, quais sugestões você daria para a melhoria da ferramenta?

Juntos, os dois grupos de perguntas fornecem subsídios importantes para a validação da abordagem desenvolvida, sobretudo porque o primeiro grupo de perguntas tem o propósito de assegurar o nível de conhecimento das pessoas que responderão a pesquisa na área de desenvolvimento *mobile*, logo a aceitação de suas avaliações serão recebidas com segurança.

6.3 Seleção dos Participantes

Para que se tenha uma amostra válida de perguntados, é necessário definir as pessoas para quem o questionário é voltado. Para definir uma amostra dessa população alvo existem dois métodos, segundo Pfleeger e Kitchenham (2002): probabilístico e não-probabilístico. O método probabilístico é aplicado nos casos em que todos os membros da população pesquisada são conhecidos e apresentam uma probabilidade maior que zero para serem incluídos na amostra. Por outro lado, no método não-probabilístico os participantes da pesquisa são incluídos na amostra porque são facilmente acessíveis ou os pesquisadores entendem que eles são representativos para a população alvo.

O método não-probabilístico foi o escolhido. Assim, neste trabalho, o questionário de pesquisa foi divulgado abertamente nas mesmas listas utilizadas na *survey* do capítulo 4. Estas são listas voltadas essencialmente para desenvolvedores com conhecimento em desenvolvimento *mobile*. Seguem as listas:

- iOSDevBrasil;
- iOSDev-br;
- AndroidDev-br;
- Android-Brasil.

6.4 Metodologia para Análise

As respostas seguiram uma escala de influência que vai de 1 - Pouquíssimo Importante, passa por 3 - Importante e vai até 5 - Extremamente Importante. A análise de validade foi feita considerando o número de respostas, que variou de Importante a Extremamente Importante como positivas, e, portanto, indicaram que a solução desenvolvida neste trabalho seria válida. Por outro lado, as respostas indicadas como abaixo de Importante sugeriram que a solução não seria válida. Dessa forma, a validade foi verificada após a análise de qual dos dois vieses obteve a grande maioria das respostas.

6.5 Análise dos Resultados

O questionário ficou disponível na internet para que os participantes respondessem às perguntas por um período de 1 mês. Após esse período foram obtidas 29 respostas.

Com o intuito de verificar se as respostas foram dadas por profissionais de tecnologia da informação especializados em desenvolvimento mobile, os participantes da pesquisa foram perguntados acerca da situação profissional, idade, escolaridade e conhecimento em desenvolvimento móvel. A idade dos participantes variou de 22 a 39 anos, como ilustrado na Figura 25, 100% deles trabalham e possuem formação acadêmica em nível de graduação, especialização e mestrado, como pode ser visto nas Figuras 26 e 27, e por fim 86,3% (27.6% - Razoável; 34,5% - Alto Conhecimento; 24,1% - Muito Alto Conhecimento) declararam ter conhecimento em desenvolvimento móvel, ver Figura 28. Neste contexto, as características desta amostra revelaram que o público que respondeu à pesquisa é composto por pessoas que possuem ensino superior, estão empregadas e possuem um bom conhecimento em desenvolvimento de aplicativos *mobile*. Dessa forma, é possível assumir que eles possuem maturidade suficiente para avaliar o trabalho desenvolvido.

Figura 25 – Idade.

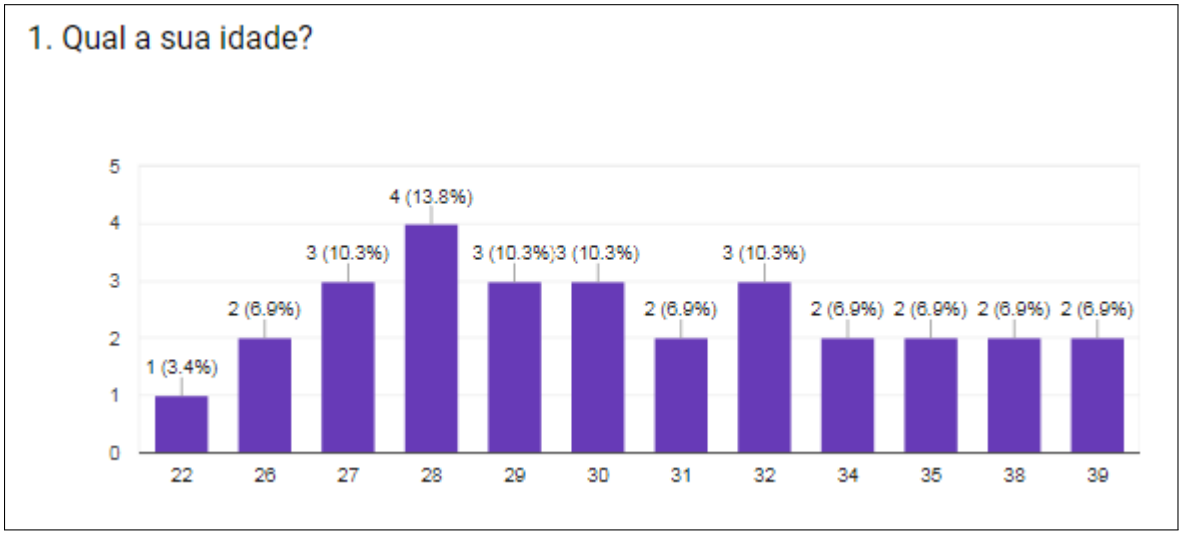


Figura 26 – Situação Profissional.

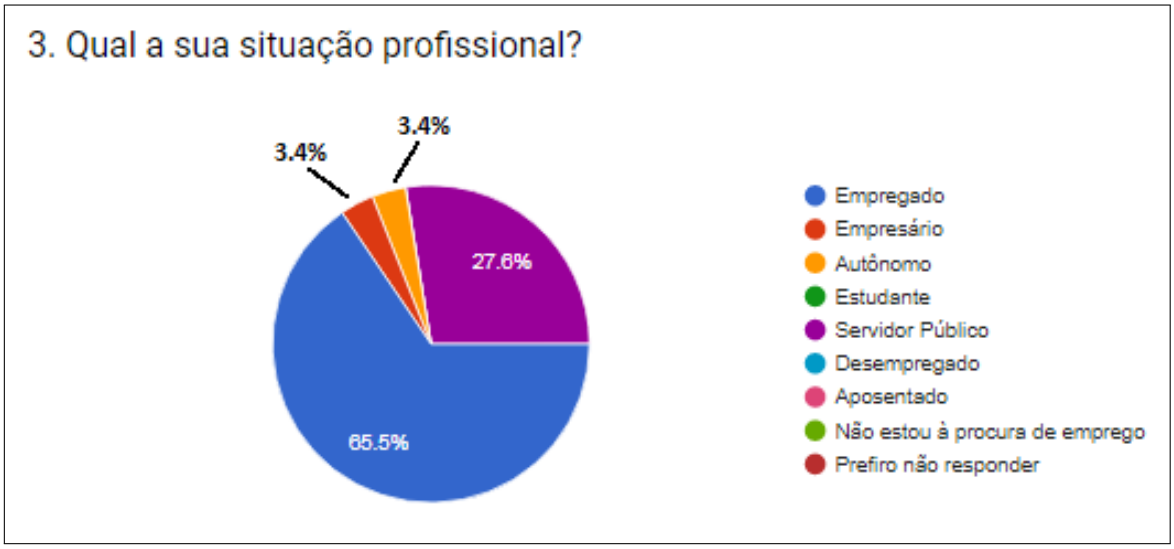


Figura 27 – Escolaridade.

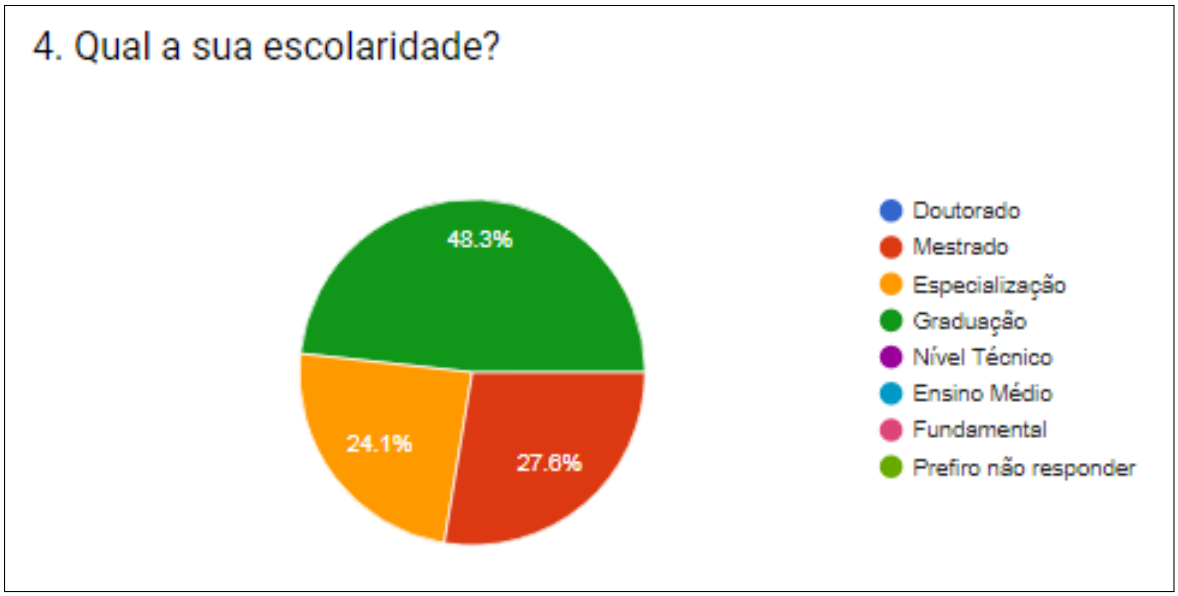
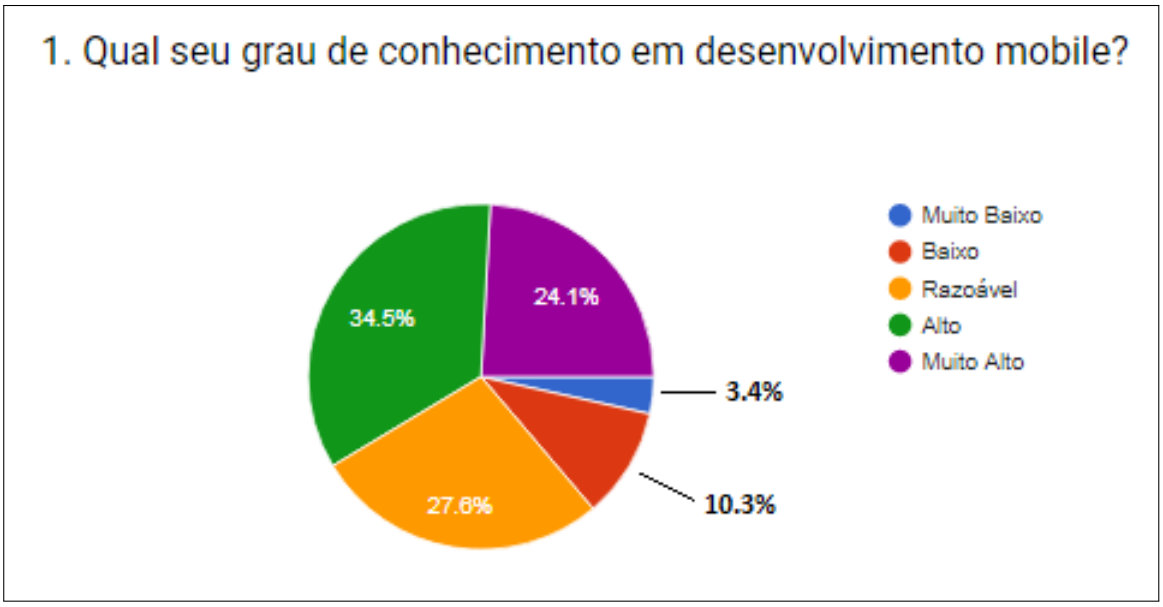


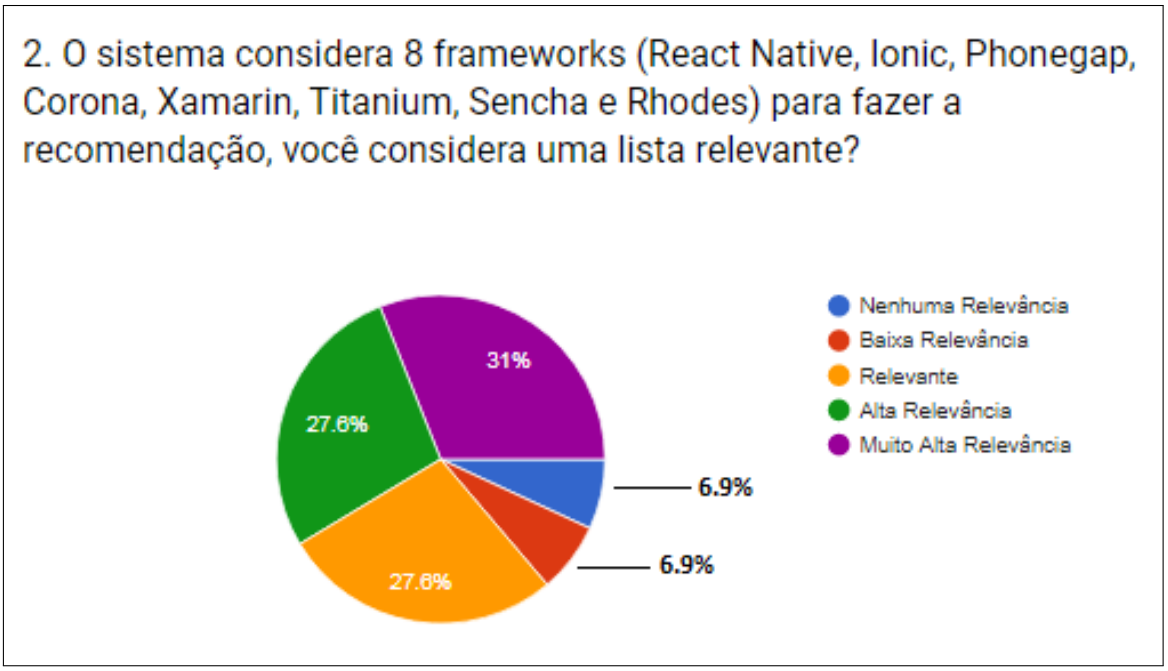
Figura 28 – Conhecimento em desenvolvimento *mobile*.



Discutiu-se com os participantes da pesquisa acerca da lista dos 8 *frameworks mobile* multiplataforma previamente escolhidos, a fim de confirmar diante da comunidade desenvolvedora de aplicativos móveis, se eles são, de fato, os principais utilizados pelo mercado. Como pode ser visto na Figura 29, 86,2% (27.6% - Relevante; 27.6% - Alta Relevância; 31% - Muito Alta Relevância) concordam com a lista. Os participantes que entenderam diferente da maioria, o fizeram não por condenar os *frameworks* selecionados, mas porque, no entendimento deles, outros deveriam ser considerados, como: Flutter[2], Kendo UI[3], Framework 7[4], NativeScript[5], Quasar[6] e Intel XDK[7]. O Flutter, *framework* lançado recentemente pelo Google foi o mais sugerido, demonstrando ter bastante popularidade entre os desenvolvedores. Por esse motivo ele foi adicionado ao sistema de recomendação, logo passou a fazer parte da lista dos principais frameworks mobile multiplataforma. Quanto aos demais frameworks suas ausências podem ser facilmente sanadas através da inclusão de suas características no banco de dados, ou seja, do ponto de vista da ferramenta é algo simples. Porém, requer um estudo aprofundado sobre os frameworks.

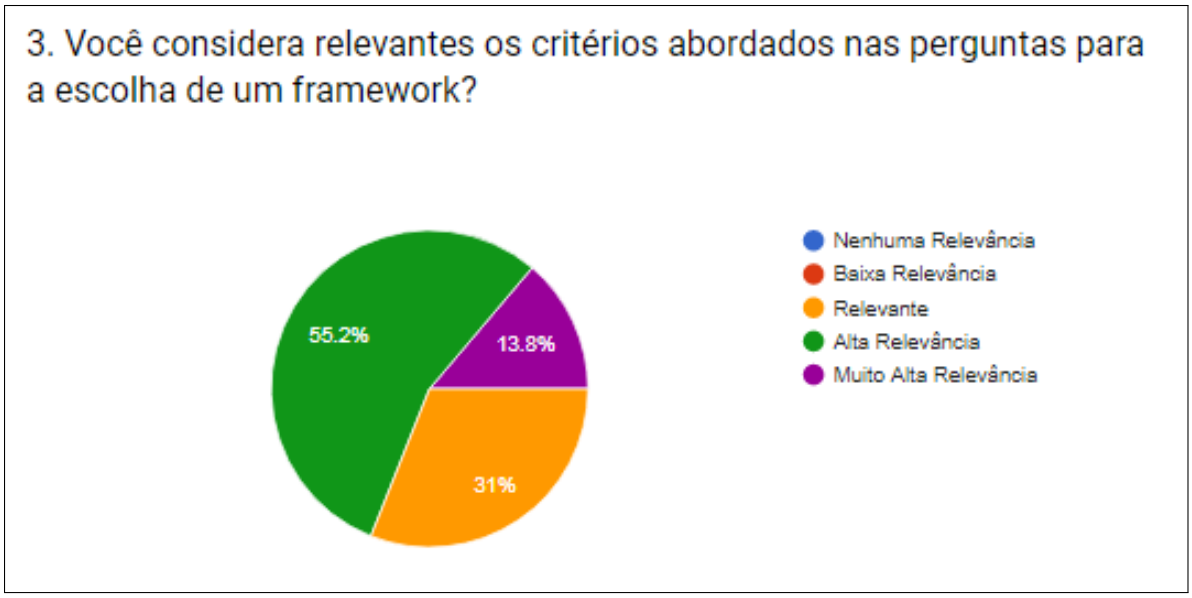
[2] https://flutter.io/
[3] https://www.telerik.com/kendo-ui
[4] https://framework7.io/
[5] https://www.nativescript.org/
[6] https://quasar-framework.org/
[7] https://software.intel.com/pt-br/xdk/article/getting-started-with-app-framework-20

Figura 29 – Relevância dos Frameworks.



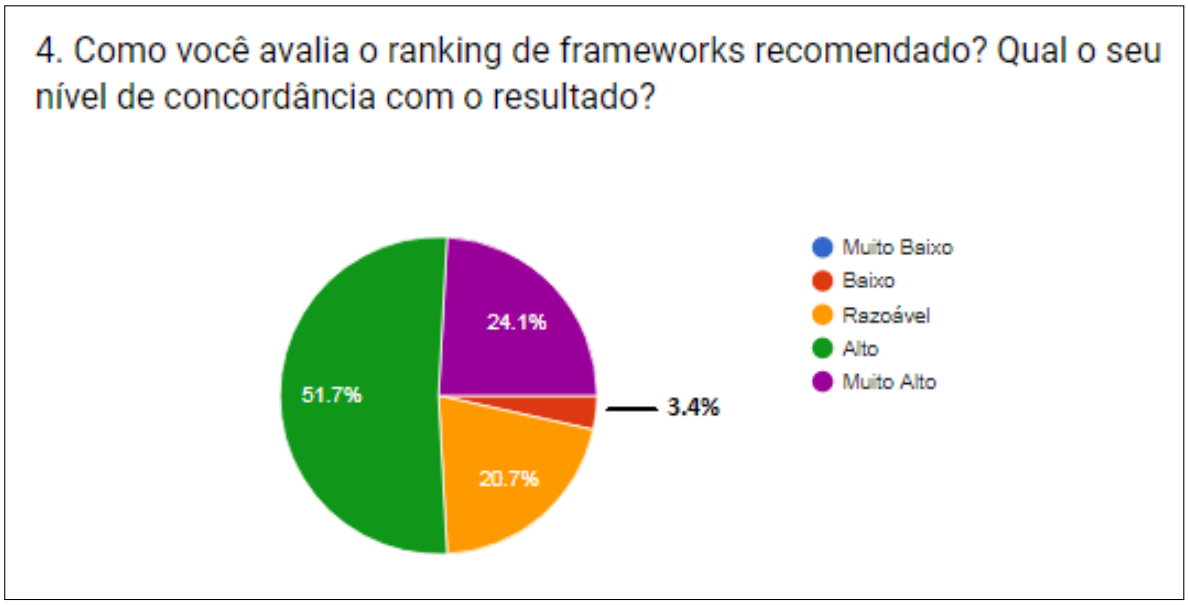
Os critérios de relevância definidos através da pesquisa discutida no capítulo 4 foram novamente abordados e confirmados por 100% (31% - Relevante; 55,2% - Alta Relevância; 13,8% - Muito Alta Relevância) dos participantes, como ilustrado na Figura 30. Mas, segundo eles, três novos critérios deveriam ser considerados: quantidade de desenvolvedores no mercado, performance e nível de produtividade. A quantidade de desenvolvedores pode ser obtida através do site de cada *framework*, como é o caso do Titanium, ou através da quantidade de membros das principais comunidades especializadas. Quanto à performance e ao nível de produtividade faz-se necessário, através de um estudo mais aprofundado, criar métricas para que seja possível classificar um *framework* em uma escala de performance e outra de produtividade.

Figura 30 – Relevância dos Critérios.



A abordagem desenvolvida entrega um ranking onde a primeira posição é ocupada pelo *framework* mais adequado às necessidades do desenvolvedor e a última o menos adequado, ou seja, ele segue uma ordem decrescente de adequação. Como explicado no capítulo 5, o ranking é montado considerando os critérios definidos no capítulo 4 e a lista dos principais *frameworks*. Esse ranking também foi avaliado pelos desenvolvedores que fizeram uso da aplicação. Para 96,5% (20,7% - Razoável; 51,7% - Alto; 24,1% - Muito Alto) deles o ranking está em conformidade com o que eles esperavam, como pode ser visto na Figura 31.

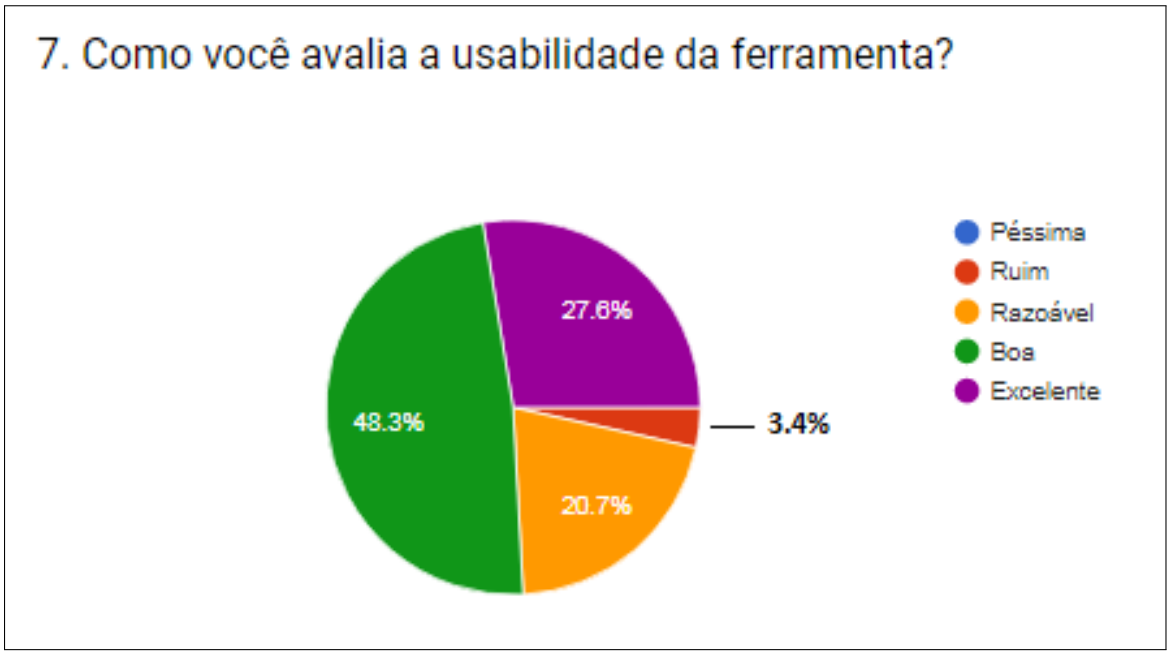
Figura 31 – Relevância do Ranking.



A usabilidade da aplicação também foi um ponto abordado nessa pesquisa avaliativa. Como ilustrado na Figura 32, para 96,6% (20,7% - Razoável; 51,7% - Boa; 24,1% - Excelente) a

aplicação se apresenta de forma simples e intuitiva quanto a seu uso, mas para 3,4% a usabilidade é ruim.

Figura 32 – Usabilidade da Ferramenta.



A pesquisa buscou também perguntar sobre a confiança que a aplicação mostrou ter. Como pode ser visto na Figura 33, 86,2% (34,5% - Relevante; 31% - Alta Relevância; 20,7% - Muito Alta Relevância) consideram a ferramenta relevante para o auxílio da escolha de um *framework mobile* multiplataforma. Além disso, 86,2% (34,5% - Provavelmente; 20,7% - Alta; 31% - Muito Alta) recomendariam a aplicação para um outro desenvolvedor ou empresa, ver Figura 34. Apenas uma pessoa respondeu que não indicaria a ferramenta, mas ele justificou que não o faria pois somente acredita em desenvolvimento móvel nativo.

Figura 33 – Relevância da Ferramenta.

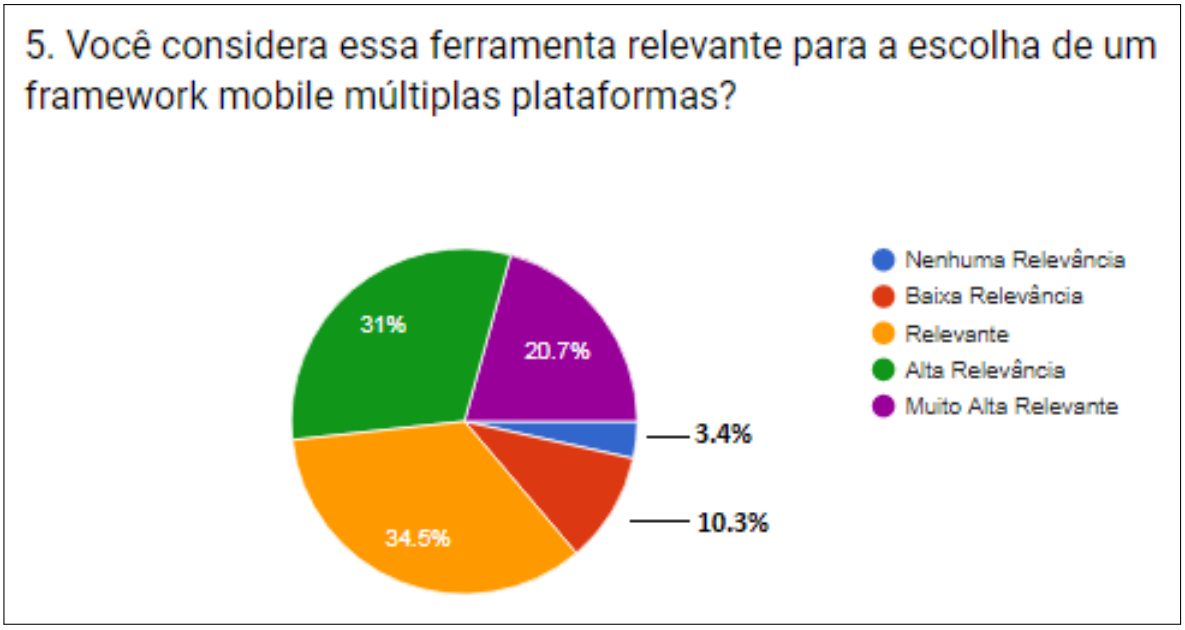
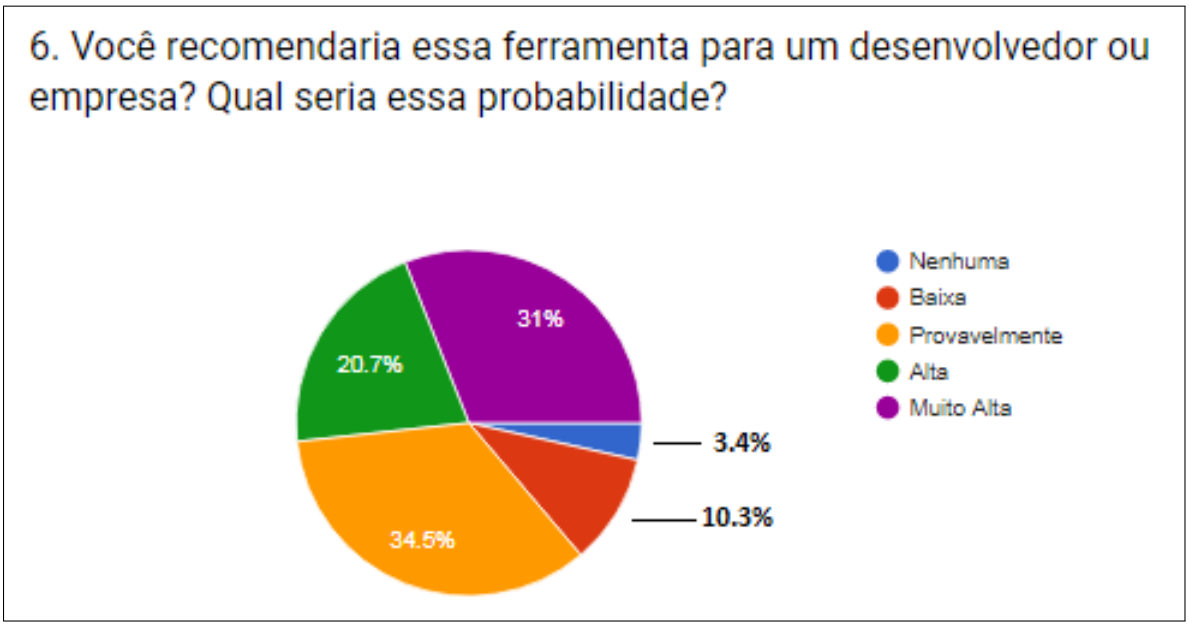


Figura 34 – Indicação da Ferramenta a Empresas.



Os participantes da pesquisa tiveram a oportunidade de escrever algumas observações, pois no questionário possuía uma questão discursiva. Essa questão teve por objetivo deixar os envolvidos na pesquisa livres para abordar temas não discutidos na pesquisa, críticas e elogios. Como resultado, foram obtidos alguns elogios parabenizando a contribuição para a área de desenvolvimento móvel tão crescente atualmente. Também foi sugerida a adição de mais *frameworks*, considerados por eles, importantes no mercado. Foi sugerido também ampliar a quantidade de recursos suportados, tais como: mapas e uso da câmera. Duas pessoas entenderam que a ferramenta é de grande relevância para desenvolvedores ou empresas que desejam iniciar nesse mundo multiplataforma.

Após observar os percentuais apresentados, todos positivos, onde a grande maioria entendeu que a lista de *frameworks*, os critérios abordados e o *ranking* apresentaram-se de forma efetiva, é possível entender que a abordagem desenvolvida obteve um bom retorno da comunidade de desenvolvimento móvel, e, assim, atingiu os objetivos esperados. Logo, a hipótese definida no capítulo 1, seção 1.3 é verdadeira.

6.6 Considerações Finais do Capítulo

Neste capítulo foi apresentada a validação da abordagem apresentada no capítulo 5. Ela foi realizada através de uma pesquisa realizada entre desenvolvedores de aplicativos móveis. Os resultados foram bastante satisfatórios, os participantes da pesquisa avaliaram positivamente todos os itens perguntados. Dessa forma, pode-se concluir que a abordagem foi efetiva.

No próximo capítulo serão apresentadas as conclusões deste trabalho, principais contribuições, limitações da pesquisa e trabalhos futuros.

7
Conclusão

Neste trabalho discutiu-se a identificação dos principais *frameworks mobile* multiplataforma utilizados no mercado pelos desenvolvedores mobile. Foi realizada também uma pesquisa em listas de discussão especializadas em desenvolvimento mobile que objetivou identificar quais fatores devem ser considerados na escolha de um *framework*. A partir dessas pesquisas foi desenvolvida uma abordagem a fim de recomendar qual *framework* é o mais indicado para um dado desenvolvedor e seu projeto. Essa abordagem foi implementada e disponibilizada publicamente. Foi realizada uma divulgação da abordagem em listas de discussão, compostas por especialistas em desenvolvimento mobile, solicitando que após experimentar a ferramenta fosse respondido um questionário de validação para que os especialistas analisassem e validassem a abordagem desenvolvida, a lista de *frameworks* apresentada e os fatores considerados.

Os resultados da pesquisa mostraram que a comunidade de desenvolvimento *mobile* recebeu bem a solução apresentada. Foi possível chegar a essa conclusão com base nos resultados da análise apresentada no capítulo anterior, onde as porcentagens de aprovação dos itens perguntados foram todas acima de 85%, com três destaques: quando o questionamento procurou descobrir a satisfação do participante da pesquisa em relação à lista de *frameworks* selecionados, 86,2% responderam positivamente; quando o assunto abordou os critérios que devem ser utilizados na escolha de um *framework*, 100% foi a porcentagem de satisfação; e por fim, quando interpelou-se sobre o ranking, 96% julgaram-no correto. Sendo assim, conclui-se que através da abordagem desenvolvida será possível fornecer auxílio aos desenvolvedores na escolha de um *framework mobile* multiplataforma, poupando-lhes tempo na procura entre as várias opções existentes no mercado.

7.1 Principais Contribuições

As principais contribuições deste trabalho foram:

- Definição de fatores relevantes que devem ser analisados quando se desejar escolher um *framework mobile* multiplataforma mais adequado às necessidades do desenvolvedor.
- Identificação dos principais *frameworks mobile* multiplataforma utilizados no mercado pelos desenvolvedores de aplicativos móveis.
- Criação de uma abordagem que permite ao desenvolvedor ter o *framework mobile* multiplataforma mais adequado ao seu projeto.
- Criação de um sistema de recomendação que implementa a abordagem desenvolvida.

7.2 Limitações da Pesquisa

Uma limitação deste trabalho refere-se à *survey* apresentada no capítulo 6. Ela teve o objetivo de avaliar a abordagem desenvolvida perante a comunidade desenvolvedora de aplicativos móveis. Mas a pesquisa foi realizada somente em comunidade brasileiras. Como os *frameworks* são utilizados por desenvolvedores que estão espalhados por todo o mundo, seria importante saber a opinião dos estrangeiros em relação aos critérios de escolha e *frameworks* adotados, tanto quanto a efetividade da abordagem desenvolvida. A percepção dos estrangeiros pode ser diferente em relação à dos brasileiros.

7.3 Trabalhos futuros

Para os futuros trabalhos sugere-se que o fator linguagem de programação, considerado como relevante para escolha de um *framework mobile* multiplataforma, seja trabalhado de maneira mais detalhada. Na solução desenvolvida são listadas somente as linguagens suportadas pelos *frameworks* para o desenvolvedor, mas poderia ser realizado um trabalho a fim de identificar semelhanças entre as linguagens e, dessa forma, seria possível aumentar as opções apresentadas para quem faz uso do sistema desenvolvido. Por exemplo, se for identificado que C# é semelhante a Java, então a recomendação do framework será mais provável. A intenção é aumentar as possibilidades para o desenvolvedor que não conhece nenhuma linguagem de programação suportada pelos *frameworks* trabalhados, mas conhece alguma semelhante.

Sugere-se também como contribuição para melhoria do conjunto de critérios de escolha de um *frameworks mobile* multiplataforma a adição de três novos itens sugeridos pelos desenvolvedores no capítulo 6: performance, quantidade de desenvolvedores no mercado e nível de produtividade. A quantidade de desenvolvedores pode ser obtida de forma direta através do site de cada *framework* ou através da quantidade de membros das principais comunidades especializadas. Quanto à performance e ao nível de produtividade faz-se necessário, através de um estudo mais aprofundado, criar métricas para que se tenha a possibilidade de classificar um *framework* em uma escala de performance e outra de produtividade. Dessa forma, será

possível escolher a medida de similaridade mais adequada e, por fim, atualizar o sistema de recomendação.

Referências

AGGARWAL, C. C. Knowledge-based recommender systems. In: ____. *Recommender Systems: The Textbook*. Cham: Springer International Publishing, 2016. p. 167–197. ISBN 978-3-319-29659-3. Disponível em: <http://dx.doi.org/10.1007/978-3-319-29659-3_5>. Citado 3 vezes nas páginas 34, 35 e 38.

AGGARWAL, C. C. *Recommender Systems: The Textbook*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2016. ISBN 3319296574, 9783319296579. Citado 3 vezes nas páginas 31, 32 e 33.

BERNARDES, T. F.; MIYAKE, M. Y. Cross-platform mobile development approaches: A systematic review. *IEEE Latin America Transactions*, v. 14, n. 4, p. 1892–1898, April 2016. ISSN 1548-0992. Citado 4 vezes nas páginas 19, 21, 22 e 37.

BOUSHEHRINEJADMORADI, N. et al. Testing cross-platform mobile app development frameworks (t). In: *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. [S.l.: s.n.], 2015. p. 441–451. Citado na página 21.

BURKE, R. Knowledge-based recommender systems. In: *ENCYCLOPEDIA OF LIBRARY AND INFORMATION SYSTEMS*. [S.l.]: Marcel Dekker, 2000. p. 2000. Citado 2 vezes nas páginas 32 e 38.

CHA, S.-H. Comprehensive survey on distance/similarity measures between probability density functions. *City*, v. 1, n. 2, p. 1, 2007. Citado 3 vezes nas páginas 34, 35 e 38.

C.P, R. R.; TOLETY, S. B. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In: *India Conference (INDICON)*. [S.l.: s.n.], 2012. Citado na página 19.

EL-KASSAS, W. S. et al. Taxonomy of cross-platform mobile applications development approaches. *Ain Shams Engineering Journal*, v. 8, n. 2, p. 163 – 190, 2017. ISSN 2090-4479. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2090447915001276>. Citado 2 vezes nas páginas 17 e 20.

FOWLER, M. *Developing Software for Multiple Mobile Devices*. 2012. Disponível em: <http://martinfowler.com/articles/multiMobile/>. Citado na página 18.

FURLAN, A. *Cross Platform Mobile App Development Guide (2017)*. 2017. Http://www.businessofapps.com/guide/cross-platform-mobile-app-development/. Citado 2 vezes nas páginas 22 e 23.

GIASSON, G. luis da S. *9 Frameworks de Desenvolvimento Multiplataforma Móvel*. 2017. Http://micreiros.com/9-frameworks-de-desenvolvimento-multiplataforma-movel/. Citado 2 vezes nas páginas 22 e 23.

GOKHALE, P.; SINGH, S. Multi-platform strategies, approaches and challenges for developing mobile applications. In: *Circuits, Systems, Communication and Information Technology Applications (CSCITA), 2014 International Conference on*. [S.l.: s.n.], 2014. p. 289–293. Citado 3 vezes nas páginas 12, 13 e 19.

(IDC), I. D. C. *Smartphone OS Market Share, 2017 Q1*. 2017. Disponível em: <http://www.idc.com/promo/smartphone-market-share/os>. Citado na página 12.

JAIN, S. et al. Trends, problems and solutions of recommender system. In: *International Conference on Computing, Communication Automation*. [S.l.: s.n.], 2015. p. 955–958. Citado 3 vezes nas páginas 35, 36 e 37.

JSCRAMBLER. *10 Frameworks for Mobile Hybrid Apps*. 2017. Https://blog.jscrambler.com/10-frameworks-for-mobile-hybrid-apps. Citado 2 vezes nas páginas 22 e 23.

KHAN, S. *5 Best Cross Platform Mobile Development Tools*. 2015. Http://www.cygnismedia.com/blog/best-mobile-development-tools/. Citado 2 vezes nas páginas 22 e 23.

KUMAR, A.; TANWAR, P.; NIGAM, S. Survey and evaluation of food recommendation systems and techniques. In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. [S.l.: s.n.], 2016. p. 3592–3596. Citado na página 31.

LATIF, M. et al. Cross platform approach for mobile application development: A survey. In: *2016 International Conference on Information Technology for Organizations Development (IT4OD)*. [S.l.: s.n.], 2016. p. 1–5. Citado 4 vezes nas páginas 13, 18, 20 e 22.

LAVETI, R. N. et al. A hybrid recommender system using weighted ensemble similarity metrics and digital filters. In: *2016 IEEE 23rd International Conference on High Performance Computing Workshops (HiPCW)*. [S.l.: s.n.], 2016. p. 32–38. Citado na página 35.

LOPS, M. d. G. P.; SEMERARO, G. *Content-based recommender systems: State of the art and trends*. 2011. Disponível em: <http://www.ics.uci.edu/~welling/teaching/CS77Bwinter12/handbook/ContentBasedRS.pdf>. Citado na página 32.

MARKOV, D. *Comparing The Top Frameworks For Building Hybrid Mobile Apps*. 2015. Disponível em: <http://tutorialzine.com/2015/10/comparing-the-top-frameworks-for-building-hybrid-mobile-apps/>. Citado na página 13.

MEDIUM. *Cross-platform Frameworks for Mobile Development*. 2017. Https://medium.com/@MasterOfCodeGlobal/best-10-android-frameworks-for-building-android-apps-d2d0ee48e464. Citado 2 vezes nas páginas 22 e 23.

PALMIERI, M.; SINGH, I.; CICCHETTI, A. Comparison of cross-platform mobile development tools. In: *16th International Conference on Intelligence in Next Generation Networks (ICIN)*. [S.l.: s.n.], 2012. Citado na página 37.

PFLEEGER, S. L.; KITCHENHAM, B. A. Principles of survey research: Part 1: Turning lemons into lemonade. *SIGSOFT Softw. Eng. Notes*, ACM, New York, NY, USA, v. 26, n. 6, p. 16–18, nov. 2001. ISSN 0163-5948. Disponível em: <http://doi.acm.org/10.1145/505532.505535>. Citado na página 40.

PFLEEGER, S. L.; KITCHENHAM, B. A. Principles of survey research: Part 5: Populations and samples. *SIGSOFT Softw. Eng. Notes*, ACM, New York, NY, USA, v. 27, n. 5, p. 17–20, nov. 2002. ISSN 0163-5948. Disponível em: <http://doi.acm.org/10.1145/571681.571686>. Citado 2 vezes nas páginas 42 e 69.

REDBYTES. *9 Best Cross-Platform Mobile App Development Tools*. 2017. Https://www.redbytes.in/9-best-cross-platform-mobile-app-development-tools/. Citado 2 vezes nas páginas 22 e 23.

SILVA, E. L.; MENEZES, E. M. *Metodologia da Pesquisa e Elaboração de Dissertação*. Santa Catarina: Universidade Federal de Santa Catarina, 2001. Citado na página 15.

SINGAL, S. M.; TEJAL; JUNEJA, B. Adaboosting for case-based recommendation system. In: *2016 International Conference on Information Technology (InCITe) - The Next Generation IT Summit on the Theme - Internet of Things: Connect your Worlds*. [S.l.: s.n.], 2016. p. 62–66. Citado na página 34.

SINGH, A. K. *Top 10 Cross-Platform Mobile Development Tools*. 2017. Https://www.hongkiat.com/blog/cross-mobile-platform-framework-wora/. Citado 2 vezes nas páginas 22 e 23.

TIWARI, S. *5 Best Cross-Platform Mobile App Development Tools*. 2016. Disponível em: <https://www.techaheadcorp.com/blog/mobile-applications/5-best-cross-platform-mobile-app-development-tools/>. Citado 2 vezes nas páginas 22 e 23.

VENTURA, M. M. *O Estudo de Caso como Modalidade de Pesquisa*. [S.l.]: Pedagogia Médica, 2007. Citado na página 15.

VOLLRATH, I.; WILKE, W.; BERGMANN, R. Case-based reasoning support for online catalog sales. *IEEE Internet Computing*, v. 2, n. 4, p. 47–54, Jul 1998. ISSN 1089-7801. Citado na página 34.

Apêndices

APÊNDICE A – Questionário de Pesquisa

Este questionário foi criado e disponibilizado em meio eletrônico, com o objetivo de coletar dados a fim de investigar sobre os fatores que influenciam no processo de adoção de *frameworks* de desenvolvimento *mobile* multiplataforma. A seguir são apresentadas todas as perguntas e respostas.

Você possui alguma experiência com **desenvolvimento de aplicativos** para dispositivos móveis?

136 de 136 pessoas responderam esta pergunta

Média: 3.52

1	2	3	4	5
Nenhuma		Pouca		Sou Expert

4		60 / **44%**
3		30 / **22%**
5		23 / **17%**
1		12 / **9%**
2		11 / **8%**

Quais linguagens de programação você conhece?

136 de 136 pessoas responderam esta pergunta

1	Java	109 / **80%**
2	HTML	94 / **69%**
3	Javascript	89 / **65%**
4	CSS	80 / **59%**
5	C#	66 / **49%**
6	Swift	48 / **35%**
7	Objective-C	47 / **35%**
8	Python	41 / **30%**
9	Ruby	24 / **18%**
10	Other	18 / **13%**

Legal que você conhece **Java**! Qual seu nível de conhecimento na linguagem?

94 de 136 pessoas responderam esta pergunta

Média: 3.56

1	2	3	4	5

Pouquíssimo Conhecimento — Conhecimento Médio — Sou Expert

4	39 / **41%**
3	35 / **37%**
5	12 / **13%**
2	6 / **6%**
1	2 / **2%**

Legal que você conhece **Swift**! Qual seu nível de conhecimento na linguagem?

47 de 136 pessoas responderam esta pergunta

Média: 3.43

1	2	3	4	5

Pouquíssimo Conhecimento — Conhecimento Médio — Sou Expert

4	17 / **36%**
3	15 / **32%**
2	9 / **19%**
5	6 / **13%**

Legal que você conhece **Objective-C**! Qual seu nível de conhecimento na linguagem?

43 de 136 pessoas responderam esta pergunta

Média: 3.77

1	2	3	4	5

Pouquíssimo Conhecimento — Conhecimento Médio — Sou Expert

5	16 / **37%**
3	10 / **23%**

4	10 / **23%**
2	5 / **12%**
1	2 / **5%**

Legal que você conhece **Python**! Qual seu nível de conhecimento na linguagem?

33 de 136 pessoas responderam esta pergunta

Média: 3.18

1	2	3	4	5

Pouquíssimo Conhecimento — Conhecimento Médio — Sou Expert

3	13 / **39%**
4	10 / **30%**
2	8 / **24%**
5	2 / **6%**

Legal que você conhece **Ruby**! Qual seu nível de conhecimento na linguagem?

20 de 136 pessoas responderam esta pergunta

Média: 2.90

1	2	3	4	5

Pouquíssimo Conhecimento — Conhecimento Médio — Sou Expert

2	8 / **40%**
3	6 / **30%**
4	6 / **30%**

Legal que você conhece **C#**! Qual seu nível de conhecimento na linguagem?

60 de 136 pessoas responderam esta pergunta

Média: 3.42

1	2	3	4	5

Pouquíssimo Conhecimento Conhecimento Médio Sou Expert

4	28 / **47%**
3	17 / **28%**
2	7 / **12%**
5	5 / **8%**
1	3 / **5%**

Legal que você conhece **Javascript**! Qual seu nível de conhecimento na linguagem?

83 de 136 pessoas responderam esta pergunta

Média: 3.59

1	2	3	4	5

Pouquíssimo Conhecimento Conhecimento Médio Sou Expert

4	37 / **45%**
3	28 / **34%**
5	10 / **12%**
2	8 / **10%**

Legal que você conhece **HTML**! Qual seu nível de conhecimento na linguagem?

83 de 136 pessoas responderam esta pergunta

Média: 4.00

1	2	3	4	5

Pouquíssimo Conhecimento Conhecimento Médio Sou Expert

4	36 / **43%**
5	25 / **30%**
3	19 / **23%**
2	3 / **4%**

Legal que você conhece **CSS**! Qual seu nível de conhecimento na linguagem?

74 de 136 pessoas responderam esta pergunta

Média: 3.49

1	2	3	4	5
Pouquíssimo Conhecimento		Conhecimento Médio		Sou Expert

3		31 / **42%**
4		22 / **30%**
5		12 / **16%**
2		8 / **11%**
1		1 / **1%**

Você estaria disposto a aprender uma nova linguagem de programação para poder desenvolver aplicativos mobile?

136 de 136 pessoas responderam esta pergunta

1	Sim	119 / **88%**
2	Não	17 / **13%**

Você (ou sua empresa) já desenvolveu algum aplicativo mobile para mais de uma plataforma (Android, iOS, etc)?

136 de 136 pessoas responderam esta pergunta

1	Sim	95 / **70%**
2	Não	41 / **30%**

Para quais **plataformas mobile** você ou sua empresa costuma desenvolver?

95 de 136 pessoas responderam esta pergunta

1	Android (Google)	93 / **98%**
2	iOS (Apple)	83 / **87%**
3	Windows Phone (Microsoft)	15 / **16%**
4	Blackberry	2 / **2%**

Você (ou sua empresa) já criou algum aplicativo utilizando algum framework de desenvolvimento mobile multiplataforma?

136 de 136 pessoas responderam esta pergunta

1	Sim	69 / **51%**
2	Não	67 / **49%**

Quais fatores te levaram (ou levariam) a escolher um framework de desenvolvimento multiplataforma?

136 de 136 pessoas responderam esta pergunta

1	Redução do custo de desenvolvimento	97 / **71%**
2	Maior Re-aproveitamento de código	94 / **69%**
3	Facilidade no design de interface	46 / **34%**
4	Conhecimento prévio da equipe numa determina linguagem	43 / **32%**
5	Facilidade na adoção de novas plataformas	43 / **32%**
6	Facilidade de adaptação	42 / **31%**
7	Melhoria na experiência do usuário	24 / **18%**
8	Maior segurança nas aplicações	12 / **9%**
9	Other	4 / **3%**

Você já conhece algum framework de desenvolvimento de aplicativos multiplataforma?

134 de 136 pessoas responderam esta pergunta

1	Sim	104 / **78%**
2	Não	30 / **22%**

Quais dos Frameworks de desenvolvimento multiplataforma abaixo você conhece?

106 de 136 pessoas responderam esta pergunta

1	Xamarin	76 / **72%**
2	Phonegap	64 / **60%**

3	Ionic	58 / **55%**
4	React Native	37 / **35%**
5	Titanium	25 / **24%**
6	Corona	19 / **18%**
7	JQuery Mobile	16 / **15%**
8	Other	12 / **11%**

Xamarin

76 de 136 pessoas responderam esta pergunta

Média: 2.16

1	2	3	4	5

Nenhum Conhecimento — Conhecimento Médio — Sou Expert

2		26 / **34%**
1		25 / **33%**
3		13 / **17%**
4		12 / **16%**

React Native

37 de 136 pessoas responderam esta pergunta

Média: 2.30

1	2	3	4	5

Nenhum Conhecimento — Conhecimento Médio — Sou Expert

2		14 / **38%**
1		9 / **24%**
3		9 / **24%**
4		4 / **11%**

5	1 / **3%**

Titanium

25 de 136 pessoas responderam esta pergunta

Média: 2.04

1	2	3	4	5
Nenhum Conhecimento		Conhecimento Médio		Sou Expert

2	12 / **48%**
1	7 / **28%**
3	4 / **16%**
4	2 / **8%**

Ionic

58 de 136 pessoas responderam esta pergunta

Média: 2.57

1	2	3	4	5
Nenhum Conhecimento		Conhecimento Médio		Sou Expert

2	16 / **28%**
3	15 / **26%**
1	13 / **22%**
4	11 / **19%**
5	3 / **5%**

Phonegap

64 de 136 pessoas responderam esta pergunta

Média: 2.42

1	2	3	4	5

Nenhum Conhecimento Conhecimento Médio Sou Expert

2	19 / **30%**
1	17 / **27%**
3	15 / **23%**
4	10 / **16%**
5	3 / **5%**

Corona

19 de 136 pessoas responderam esta pergunta

Média: 2.11

1	2	3	4	5

Nenhum Conhecimento Conhecimento Médio Sou Expert

2	8 / **42%**
1	6 / **32%**
3	3 / **16%**
4	1 / **5%**
5	1 / **5%**

JQuery Mobile

16 de 136 pessoas responderam esta pergunta

Média: 3.19

1	2	3	4	5

Nenhum Conhecimento Conhecimento Médio Sou Expert

4	5 / **31%**
2	4 / **25%**
3	4 / **25%**

5		2 / **13%**
1		1 / **6%**

Na sua opinião, quais as plataformas mais importantes do mercado?

136 de 136 pessoas responderam esta pergunta

1	Android	136 / **100%**
2	iOS	133 / **98%**
3	Windows Phone	11 / **8%**
4	Blackberry	0 / **0%**

Que tipo de abordagem de desenvolvimento mobile você considera ideal?

136 de 136 pessoas responderam esta pergunta

1	Nativa	96 / **71%**
2	Híbrida	35 / **26%**
3	Web	5 / **4%**

Quais recursos do dispositivo um framework precisa suportar para satisfazer suas necessidades atuais?

136 de 136 pessoas responderam esta pergunta

1	Rede e Dados	119 / **88%**
2	GPS e Localização	113 / **83%**
3	Armazenamento	110 / **81%**
4	Câmera	97 / **71%**
5	Leitor de impressão digital	51 / **38%**
6	Lista de Contatos	48 / **35%**
7	Acelerômetro	44 / **32%**
8	Bluetooth	40 / **29%**
9	Aceleração de Vídeo	33 / **24%**

10	NFC	29 / **21%**
11	Assistente Virtual	22 / **16%**
12	Bússola	17 / **13%**
13	Other	3 / **2%**

Para você, dos itens abaixo qual o mais importante no momento de escolher um framework?

136 de 136 pessoas responderam esta pergunta

1	Tamanho da Comunidade	56 / **41%**
2	Linguagem utilizada pelo Framework	46 / **34%**
3	Preço	24 / **18%**
4	Restrições e Licença (Apache, GPL, etc)	10 / **7%**

Você pagaria para utilizar um framework multiplataforma?

136 de 136 pessoas responderam esta pergunta

1	Não	79 / **58%**
2	Sim	57 / **42%**

O quão importante é o suporte à versão mais recente da plataforma (Android, iOS) no momento da escolha de um framework multiplataforma?

136 de 136 pessoas responderam esta pergunta

Média: 4.39

1	2	3	4	5
Pouco Importante		Importante		Extremamente Importante

5		77 / **57%**
4		38 / **28%**
3		18 / **13%**
2		3 / **2%**

Qual a sua idade?

136 de 136 pessoas responderam esta pergunta

1	25 - 34	83 / **61%**
2	19 - 24	32 / **24%**
3	35 - 44	16 / **12%**
4	45 - 54	4 / **3%**
5	Menor de 18	1 / **1%**
6	55 - 64	0 / **0%**
7	65 ou mais	0 / **0%**

Qual seu gênero?

136 de 136 pessoas responderam esta pergunta

1	Masculino	126 / **93%**
2	Feminino	8 / **6%**
3	Homem	2 / **1%**

Qual seu país de nascimento?

136 de 136 pessoas responderam esta pergunta

1	Brazil	135 / **99%**
2	India	1 / **1%**
3	Afghanistan	0 / **0%**
4	Albania	0 / **0%**
5	Algeria	0 / **0%**
6	Andorra	0 / **0%**
7	Angola	0 / **0%**
8	Antigua and Barbuda	0 / **0%**

9	Argentina	0 / **0%**
10	Armenia	0 / **0%**
11	Aruba	0 / **0%**
12	Australia	0 / **0%**
13	Austria	0 / **0%**
14	Azerbaijan	0 / **0%**
•••	Outro	0 / **0%**

Qual sua situação profissional?

136 de 136 pessoas responderam esta pergunta

1	Empregado	81 / **60%**
2	Empresário	20 / **15%**
3	Estudante	14 / **10%**
4	Servidor Público	11 / **8%**
5	Desempregado	5 / **4%**
6	Prefiro não responder	5 / **4%**
7	Aposentado	0 / **0%**
8	Não estou a procura de emprego	0 / **0%**

Qual sua escolaridade?

136 de 136 pessoas responderam esta pergunta

1	Graduação	73 / **54%**
2	Pós-graduação	31 / **23%**
3	Mestrado	11 / **8%**
4	Nível Técnico	9 / **7%**
5	Ensino Médio	8 / **6%**

6	Doutorado	2 / **1%**
7	Prefiro não responder	2 / **1%**
8	Fundamental	0 / **0%**

APÊNDICE B – Questionário de Validação de Pesquisa

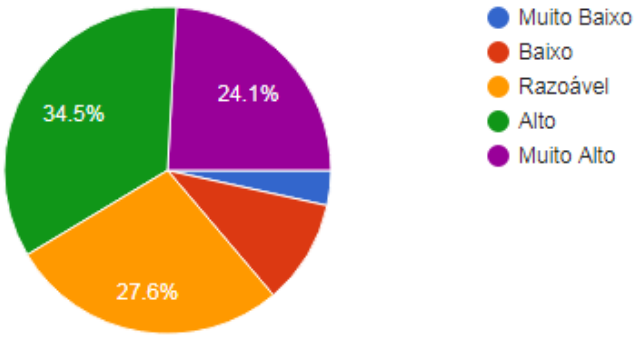
Este questionário foi criado e disponibilizado em meio eletrônico, com o objetivo de coletar dados a fim de investigar a efetividade da solução desenvolvida. A seguir são apresentadas todas as perguntas e respostas.

Avaliação do Sistema de Recomendação

29 responses

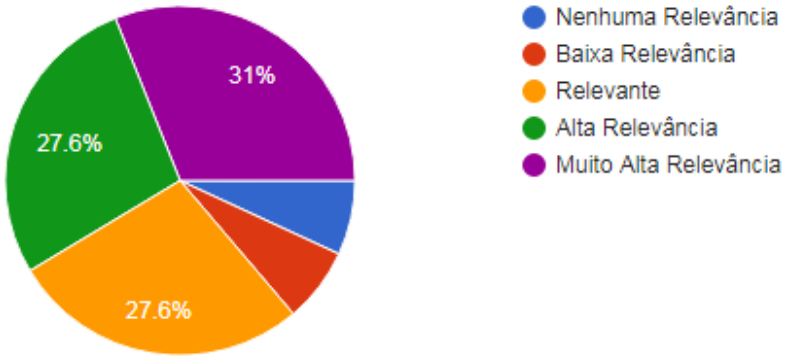
1. Qual seu grau de conhecimento em desenvolvimento mobile?

29 responses



2. O sistema considera 8 frameworks (React Native, Ionic, Phonegap, Corona, Xamarin, Titanium, Sencha e Rhodes) para fazer a recomendação, você considera uma lista relevante?

29 responses



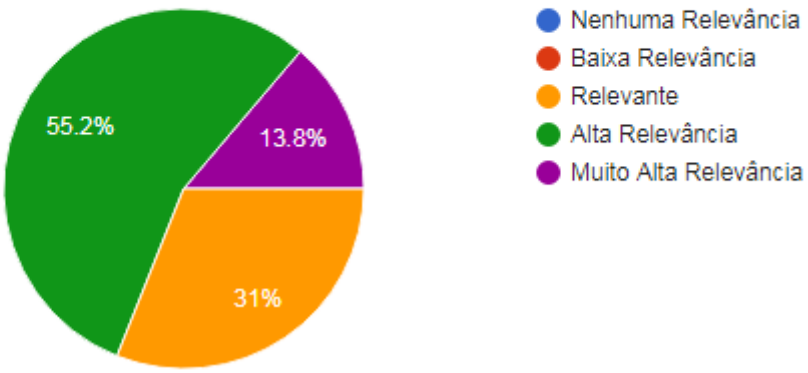
Adicionaria algum framework a lista acima? Quais?

10 responses

Flutter (2)
Kendo UI, Framework 7
NativeScript
Flutter, FuseTools, NativeScript, Weex, Quasar Framework
Intel XDK
Apesar dessa lista ser altamente relevante para mim por utilizá-las atualmente, eu aumentaria o escopo de opções, pois tenho acompanhado na comunidade Mobile a descentralização quanto a preferência de frameworks e linguagens como antigamente. Há muitas opções atualmente e os desenvolvedores ficam num vai e volta quanto a utilização deles, dependendo, por exemplo, se em um momento saiu um módulo que favoreça maior produtividade ou para outro que oferecerá maior segurança para os apps desenvolvidos. Enfim, essa lista poderia ser dinâmica, resgatando os principais frameworks a partir da API de um Web Service de uma comunidade ativa de ranking ou mesmo da API do Git. E nos informar que a lista de framework foi extraída de informações dos repositórios. Tem que ser dinâmica, um novo framework surge a cada momento, com uma curva de aprendizado baixíssima, então ele viraliza em questão de dias e milhares de apps são desenvolvidos. Não dá nem tempo de artigos científicos serem publicados. Enfim, por enquanto, eu sugiro a inclusão dos frameworks NativeScript com AngulaJS e VueJS com Weex ou Quasar.
Não conheço nenhum outro além destes.
Cordova
Não conheço muitos frameworks

3. Você considera relevantes os critérios abordados nas perguntas para a escolha de um framework?

29 responses



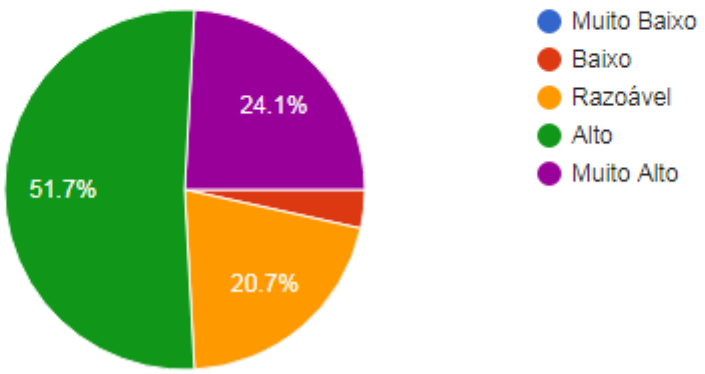
Adicionaria algum critério? Quais?

4 responses

Performance
Quantidade de desenvolvedores no mercado (alta/média/baixa), considero isso importante tanto para entender o tamanho da comunidade quanto para ter uma idéia de como é contratar um especialista nessas plataformas (concorrência)
Algum critério sobre os níveis de produtividade nas ferramentas
Não

4. Como você avalia o ranking de frameworks recomendado? Qual o seu nível de concordância com o resultado?

29 responses



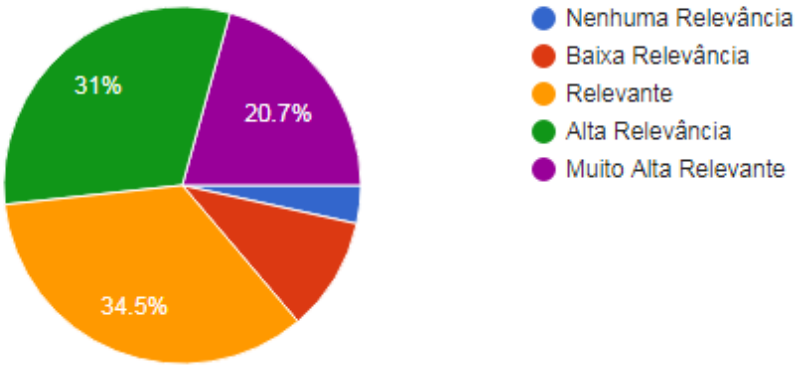
Alguma observação?

1 response

Poderia haver mais frameworks.

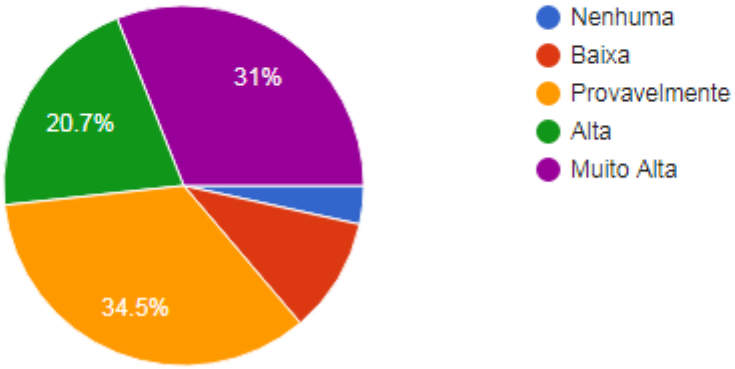
5. Você considera essa ferramenta relevante para a escolha de um framework mobile múltiplas plataformas?

29 responses



6. Você recomendaria essa ferramenta para um desenvolvedor ou empresa? Qual seria essa probabilidade?

29 responses



Alguma observação?

5 responses

Falta de outros frameworks e linguagens

Uma ferramenta como essa é fundamental à tomada de decisão na escolha de um dos diversos frameworks existentes, pois há diversos fatores que devem ser considerados, não há tempo a perder e um escolha errada pode trazer prejuízos durante e pós desenvolvimento. Temos que fazer a escolha mais acertada possível e uma ferramenta assim estava faltando no mercado. Parabéns pela iniciativa!

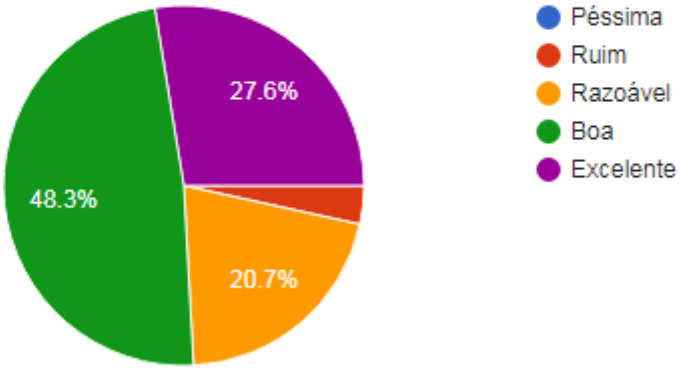
Recomendaria para uma empresa que não sabe por onde começar. Para um dev mais experiente, não vejo muito o porque recomendar, já que ele em geral terá opiniões mais formadas sobre o assunto.

Sou adepto do desenvolvimento nativo e jamais recomendaria uma solução multiplataforma por melhor que ela fosse.

Interessante para quem está interessado em começar algo e não tem conhecimento do mercado

7. Como você avalia a usabilidade da ferramenta?

29 responses



Alguma observação?

1 response

Utilizei pelo celular e quando clicava ele dava um zoom

8. De forma geral, quais sugestões você daria para a melhoria da ferramenta?

9 responses

Acrescentar outras opções de frameworks

Acho que poderiam haver outras linguagens na questão 2;

A maioria dos frameworks não enviam o app para as lojas. Eles fornecem passo a passo de como fazer;

Na questão 7, coisas como mapas, reprodução de vídeo e câmera são mais importantes que os listados;

Integração com a API do Git para resgatar os frameworks mais utilizados e compor um ranking dinamicamente ou todos os dias ou mensalmente, informando ao usuário o porque do seu framework não constar no ranking naquele momento.

Muito legal e preciso. Acho que precisa de um ajuste na usabilidade da ferramenta assim como alguns ajustes visuais para tornar o uso mais simples e agradavel

nehuma

achei meio estranho perguntar do tipo de licença com opção de selecionar apenas uma opção, se a pergunta anterior respondi que era flexível quanto a ser pago ou gratis.

Por exemplo:
Na pergunta 7, fala sobre quais recursos são relevantes para uso. Na minha opinião essa pergunta poderia não ser obrigatória, pois pode ter aplicação que nenhum daqueles recursos são relevantes e tenho que marcar ao menos 1.

No mais, acho que o questionário deviria permitir níveis de exigência do questionário. Ficando o mesmo flexível. Por exemplo, o sistema poderia me fazer mais perguntas caso queira fazer uma aplicação corporativa de grande porte, entretanto poderia me fazer menos perguntas se fosse algo mais amador ou trivial.

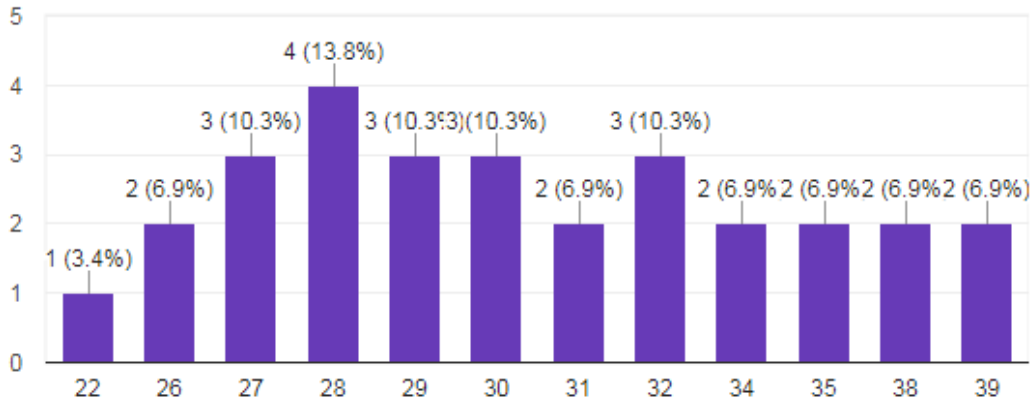
Apresentar frameworks ANTES de começar a responder se não, não fica claro que tipo de respostas terei. Adicionar Flutter.

Exibir também a alternativa marcada no resultado final (apenas mostra a relevância)

Informações Pessoais

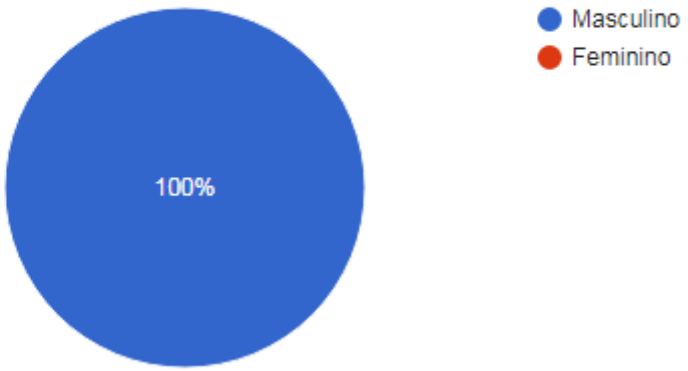
1. Qual a sua idade?

29 responses



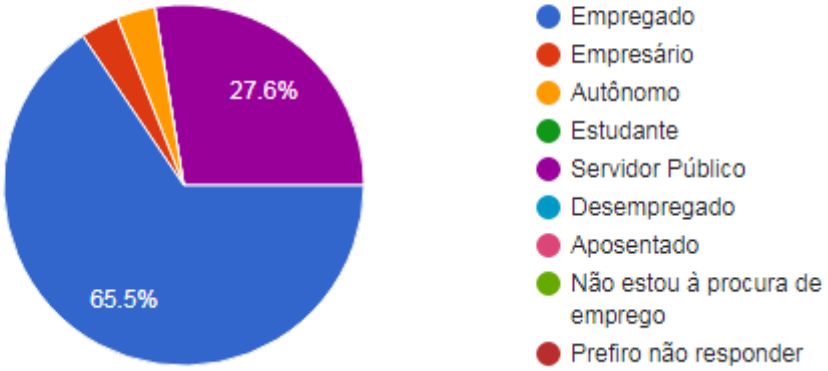
2. Qual o seu sexo?

29 responses



3. Qual a sua situação profissional?

29 responses



4. Qual a sua escolaridade?

29 responses